TECHNISCHE UNIVERSITÄT MÜNCHEN

Fakultät für Informatik

How to Speak Protein? - Representation Learning for Protein Prediction

Dissertation

Michael Heinzinger

Technische Universität München
TUM School of Computation, Information and Technology

How to Speak Protein? -
Representation Learning for Protein Prediction

Michael Heinzinger

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. Rer. Nat.)

genehmigten Dissertation.

Vorsitz: Prof. Dr. Julien Gagneur

Prüfer*innen der Dissertation:

1. Prof. Dr. Burkhard Rost
2. Prof. Dr. Martin Steinegger

Die Dissertation wurde am 23.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 05.12.2022 angenommen.

Abstract

Predicting protein function and structure from sequence is one important challenge for computational biology. For nearly three decades, most state-of-the-art approaches that took up on this challenge combined machine learning and evolutionary information (EI) condensed in multiple-sequence alignments (MSA). Despite improvements in machine learning algorithms and the way sequences in the MSA were leveraged, some major limitations inherent to EI remained. This dissertation aims at bypassing these limitations by introducing, applying, and evaluating various representation learning techniques to protein sequences. On a high-level, representation learning aims at learning meaningful vector representations (embeddings) directly from large, potentially unlabeled data without any human intervention. Those embeddings allow us to transfer the knowledge acquired by the model to any other task. Towards this end, I will use the LSTM-based language model ELMo in **Chapter 2** to showcase how concepts from natural language processing (NLP) can be adapted to protein sequences. By treating single amino acids as words and protein sequences as sentences, we could show that our model, dubbed *SeqVec*, learnt meaningful representations for protein sequences using solely unlabeled data. Crucially, all protein language models (pLMS) introduced in this dissertation rely solely on single protein sequence as input when generating embeddings. This bypasses the necessity of gathering evolutionary information (EI) explicitly by searching protein sequence databases for related proteins. Despite relying only on single protein sequences, our method *goPredSim* (**Chapter 3**) exemplifies the predictive power of the proposed representations, where we used embeddings derived from SeqVec to predict a protein's function as defined by Gene Ontology (GO) terms. In a work not described in more detail in this thesis, we improve over SeqVec by training and evaluating different Transformers on large, unlabeled sets of protein sequences. With the resulting *ProtTrans* models, we were among the first to show that embeddings derived from pLMs carry enough information to outperform complex models relying on explicit EI condensed in MSAs. The predictive power as well as the general applicability of ProtTrans embeddings is exemplified in **Chapter 4**, where we use embeddings from our best performing ProtTrans model (ProtT5) for *Variant Effect Score Prediction without Alignments* (*VESPA*). In a next step, we show how contrastive learning can refine embeddings from ProtT5 towards capturing the hierarchical classification of protein 3D structures in Class, Architecture, Topology and Homologous superfamily as defined by CATH (*ProtTucker*, **Chapter 5**). Taken together, this dissertation introduces concepts that allow to learn representations from large unlabeled databases which rival established EI-based methods in their predictive performance while relying only on single protein sequences.

Acknowledgements

First and foremost, I would like to thank Burkhard Rost for motivating me towards pursuing a PhD and inviting me to his lab. Without the great fascination for Computational Biology that you shared with me, I would never have pursued an academic path. Looking back at the insightful and inspiring discussions that we had during long walks, I can not imagine a better environment for growing not only as a scientist but also as a person.

I would also like to thank all of my colleagues from the Rostlab. Without you, I could never have accomplished what is summarized in this thesis. First, I want to thank Ahmed Elnaggar for introducing me to representation learning and Christian Dallago for many great scientific discussions on how to apply it to Computational Biology. The two of you not only know about the great results that we produced together but also about the stony path it sometimes takes to get there. - Thanks to both of you for still pursuing it! Thanks also to Maria Littmann, Céline Marquet, Konstantin Weissenow and Tobias Olenyi for all the great scientific discussions we had together and the great work that we accomplished together. Special thanks go to Inga Weise and Tim Karl for their constant support with various administrative and technical matters throughout the years. Special thanks on that regard also go to Maria Littmann not only for providing thorough feedback on my dissertation but also helping me to find my way through the bureaucratic jungle associated with it. I also want to thank all the students that I supervised throughout my PhD. It was absolutely fascinating to see you grow with your tasks. Unfortunately, I can not thank all of you explicitly so I leave it at those who are currently part of the lab: thanks Konstantin Schütze, Dagmar Ilzehöfer, Leo Kaindl and Duc Anh Le.

Special thanks go to my family, my siblings Thomas and Bettina and especially to my parents Peter and Ingrid who have been supporting me throughout my entire life. You taught me what is important in life. Without you, I would not be where I am today. I can not thank you enough for this. Thanks also to all my friends for always offering distraction from work whenever needed. The importance of this can not be overstated.

Last, but absolutely not least, I want to thank my fiancée and better half Jana Weißer for her constant support throughout my thesis. We experienced the ups and downs together and you always helped me through difficult times. Thanks for giving balance to my restless soul. Towards a bright future.

Contents

List of Figures **vii**

Abbreviations **ix**

1. Introduction **1**

1.1. Biological Background 3

1.1.1. Residue-level Structure: Secondary Structure 4

1.1.2. Protein-level Structure: CATH 4

1.1.3. Protein-level Function: Subcellular Localization 5

1.1.4. Protein-level Function: Gene Ontology 6

1.1.5. Residue-level Function: Conservation 7

1.1.6. Residue-level Function: Variant effect 7

1.2. How to Represent Data? - The Traditional Way: Handcrafted Features . 8

1.3. Leveraging Existing Data by Statistical Means Improves Representations. 9

1.4. Let Data Speak for Itself. - Learning Representations from Unlabeled Data 12

1.4.1. How to Leverage Unlabeled Data? 13

1.4.2. Context is Key - The Rise of Language Models 16

1.4.3. The Transformer Revolution 18

1.4.4. The Transformer Landscape. - A Brief Overview 21

1.4.5. Zero-shot learning and Homology-based Inference 23

1.4.6. Learning Representations from Contrasting Samples. 24

1.5. Outline of This Work 25

2. Modeling the Language of Life - SeqVec **27**

2.1. Preface 27

2.2. Journal Article: Heinzinger, Elnaggar *et al.*, BMC Bioinformatics (2019) . 28

3. Embedding-based Annotation Transfer (EAT) for GO prediction - goPredSim **47**

3.1. Preface 47

3.2. Journal Article: Littmann, Heinzinger *et al.*, Scientific Reports (2021) . . 48

4. Variant Effect Score Prediction without Alignments - VESPA **63**

4.1. Preface 63

4.2. Journal Article: Marquet, Heinzinger *et al.*, Human Genetics (2021) . . . 64

5. Contrastive Learning on Protein Embeddings Enlightens Midnight zone 85
5.1. Preface 85
5.2. Journal Article: Heinzinger *et al.*, NAR Genomics and Bioinformatics (2022) 86

6. Conclusion 105

References 113

A. Appendix 123
A.1. Publications Included in This Dissertation 125
A.1.1. Heinzinger, Elnaggar *et al.*, BMC Bioinformatics (2019) 125
A.1.2. Littmann, Heinzinger *et al.*, Nature Scientific Reports (2021) . . . 143
A.1.3. Marquet, Heinzinger *et al.*, Human Genetics (2021) 158
A.1.4. Heinzinger *et al.*, NAR Genomics and Bioinformatics (2022) 178

List of Figures

1.1. Motivation. 2
1.2. Overview of representation techniques. 14
1.3. Overview of language model pre-training and feature-extraction. 18

Abbreviations

3D	3-Dimensional
AF2	AlphaFold 2
AA	Amino Acid
ALM	Auto-regressive Language Modeling
BLOSUM	BLOcks SUbstitution Matrix
CB	Computational Biology
CNN	Convolutional Neural Network
EAT	Embedding-based Annotation Transfer
EC	Evolutionary Coupling score
EI	Evolutionary Information
FNN	Feed-forward Neural Network
GO	Gene Ontology
GPU	Graphical Processing Unit
HBI	Homology-Based Inference
HSSP	Homology-derived Secondary Structure of Proteins
HPC	High-Performance Computing
LSTM	Long Short-Term Memory
ML	Machine Learning
MLM	Masked Language Modeling
MSA	Multiple Sequence Alignment
NLP	Natural Language Processing
pLM	Protein Language Model
PSSM	Position-Specific Scoring Matrix
RNN	Recurrent Neural Network
SAV	Single Amino acid Variant
TPU	Tensor Processing Unit

1. Introduction

How do we make real world observations such as images, text, speech, or protein sequences machine-readable? - The answer to this question has changed considerably over the last decades, especially, since machine learning (ML) is increasingly leveraged in all areas of life over the last years. Among the most well-known are (nearly) self-driving cars that require real-time perception of their environment in order to react adequately to, e.g., pedestrians, other vehicles, or changing weather conditions. For this to work at an acceptable level, i.e., at equal or lower risk of accidents than human drivers, real world observations such as camera images or sensor data need to be made machine-readable at high levels of accuracy and at low computational cost to allow (nearly) real time actions. Approaching this goal relied crucially on hardware advancements. At the same time the way of reading and processing observations changed at least as fundamentally as the underlying hardware. One of the most prominent examples for this is a specific type of neural network architecture called Transformers [1] which allows to process a variety of input types. Despite being introduced in natural language processing (NLP), it not only revolutionized how we make text machine-readable [2] but also, *inter alia*, paved the way for highly accurate protein 3-dimensional (3D) structure predictions by AlphaFold 2 [3] (AF2), one of the biggest breakthroughs in computational biology (CB) [4]. Key to the inter-disciplinary applicability of the Transformer architecture is its flexibility on the expected input format which enables the network to directly process a variety of real world observations without the need for human intervention. Pairing this flexibility with clever tricks leveraging the vast amounts of unlabeled data allowed to learn data representations directly from raw data. Thanks to high-throughput sequencing technology, experimentally measuring a protein sequence is, by now, relatively cheap and fast, as opposed to measuring its 3D structure. As a result, the most common data modality available for proteins today is unlabeled raw data in the form of a consecutive string of their constituent parts, the AAs. In turn, this also means that we do not know anything about the vast majority of today's proteins but their sequence. This effect is

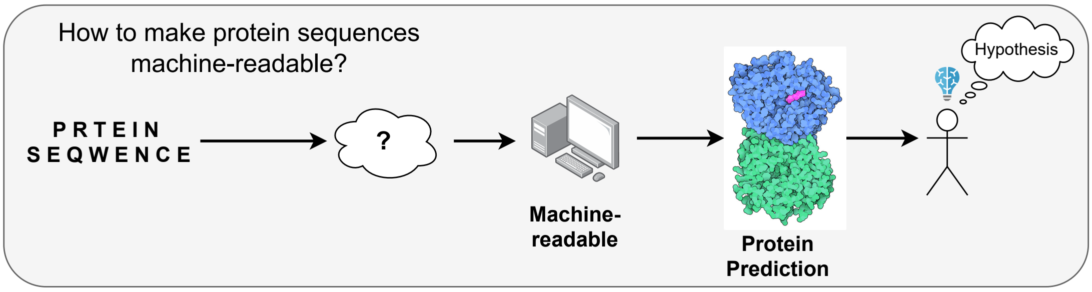

Figure 1.1.: A high-level overview of the main question that this thesis addresses: How can we make protein sequences machine-readable? We use different representations as input to machine learning devices and use their predictive performance as a proxy to answer this question. Additionally, we propose new representations learnt directly from large unlabeled protein sequence databases.

called sequence-structure gap, or more generally, sequence-annotation-gap [5]. In-silico approaches try to bridge this gap by training ML devices on the small set of proteins that have some experimental annotation (labeled data). Afterwards, the trained devices are deployed to make predictions for the huge amount of proteins without annotation (unlabeled data). However, this limits most of today's ML approaches to the rather small subset of labeled data. How much information a ML device can deduce from those small but labeled sets depends crucially on the way how its input is represented, i.e., the more informative the input, the less data is required by the model to learn something useful from it. This raises the general question on how to optimally represent protein sequences when using in-silico approaches, and, in this context, how to make best use of the constantly growing but unlabeled protein sequence databases [6, 7]. A simple overview of this problem is given in Figure 1.1. In order to provide some answers to this question, I will adapt strategies from NLP to learn representations for protein sequences and rank them by their predictive performance. In the following, I will first provide some information on the biological background followed by an overview on how previous, non-ML-based solutions processed real world observations in general and protein sequences in particular. Next, I will introduce how the proposed ML-based approaches make protein sequences machine-readable using exclusively unlabeled data. Throughout the dissertation, the proposed protein sequence representations are put into perspective of existing approaches by using predictive performance of those representations as acid test. .

1.1. Biological Background

Proteins form complex 3-dimensional (3D) macromolecules that are absolutely essential for any living organism due to their key role in nearly any cellular function. Despite their intertwined 3D appearance, proteins are composed of single amino acids (AA) that are sequentially arranged via covalent bonds. To compose this chain, 20 different AA types are available, each with unique bio-chemical and bio-physical properties defined by their corresponding side-chains. The specific arrangement of the AAs is determined by an organism's genetic blueprint that can be selectively activated based on environment factors [8]. The order of AAs and the resulting interactions between them determine the structure of a protein which in turn determines its function [9]. This can lead to residue-pairs being in close structural proximity while being far apart in sequence space. Figuratively speaking, one can imagine proteins as magnetic beads on a string where each bead resembles an AA with slightly different size, shape, polarity, electric charge, and hydrophobicity depending on the AA type. The specific arrangement of beads and the interactions between them define the resulting 3D structure. It has to be noted, that this simplification ignores many factors that complicate the physics underlying the folding process. For example, each amino acid consists of multiple atoms which allow for different interactions such as disulfid bridges or hydrogen bonds, and the folding is influenced by environment, e.g. the pH-value, in which it takes place. On top of this, and as usual in biology, there are (relatively rare) exceptions to this general rule as some proteins rely on other proteins, so-called chaperons, to adopt their final 3D fold [10]. Despite these exceptions, it is generally accepted that for the vast majority of proteins, the sequential order of AAs determines the 3D structure of a protein that in turn defines its function. This means that all information about a protein's 3D structure or function is encoded in its sequence [9]. In the following, I will give a brief overview of the different tasks that were used to evaluate different protein sequence representation. To cover a broad spectrum of the different aspects of a protein, I do not only differentiate tasks related to the structure or function of a protein but I further distinguish tasks either related to a global property of a protein (protein-level; e.g., subcellular localization) or to individual residues (residue-level; e.g., secondary structure).

1.1.1. Residue-level Structure: Secondary Structure

For most proteins, the sequential arrangement of AAs defines a protein's 3D structure. While long-range contacts between residues far apart in sequence space are crucially important for protein folding, local structural elements defined by hydrogen-bonds within their local sequence neighborhood usually drive this folding in the first place [11]. This so-called *secondary structure* can be expressed in multiple ways but commonly each residue in a protein gets assigned to one of the three following secondary structure states: helix, (beta-) strand or loop/irregular/other [12]. Helices, for example, form local patterns reminiscent of spiral staircases. Historically, secondary structure was used as a simplification to bypass the complexity of the 3D structure prediction problem while still providing some information on the structural arrangement of proteins [13–16]. Since high-quality 3D structure predictions are only a few mouse clicks away thanks to AF2, the use-cases for such a task diminished. Still, it remains one of the most well-defined and best-researched problems in CB, making it a robust benchmark for comparing different protein sequence representations (Chapter 2 and 4). From a machine-learning perspective, secondary structure prediction is a residue-level classification task, i.e., each residue in a protein gets classified into one of the three secondary structure types. Here, I used different protein sequence representations as input to different ML devices. Their predictive performance served as proxy for assessing to which extent they made protein sequences readily machine-readable. More specifically, I used the accuracy for correctly predicting secondary structure in either three- or eight-states as defined by DSSP [12] on an established dataset [16] to compare different representation types.

1.1.2. Protein-level Structure: CATH

The increasing amount of available experimental 3D structures raised the question of how to best organize this data to simplify human- and machine-readability. Towards this end, CATH [17, 18], introduced a hierarchical classification of protein 3D structures in Class, Architecture, Topology and Homologous superfamily. From the most coarse-grained Class-level where proteins only share their overall secondary structure composition, proteins are sub-divided based on increasing overlap in secondary structure up to the most fine-grained homologous superfamily-level where proteins are sequence similar. Such subdivision is essential to find evolutionary relationship among apparently unrelated proteins. It also crucially helped in quantifying that the space of possible protein 3D structures appears to be a lot smaller than the space of possible protein sequences, i.e., many proteins adopt a similar 3D structure despite having vastly diverged protein sequences [19]. This is especially true for proteins involved in essential cell functions that are shared among large fractions of the tree of life. Therefore, assigning a new protein to a CATH class does not only allow to view it against the background of the currently known structural landscape but also to derive functional hypotheses. The latter is streamlined by so-called functional families or FunFams [20] which add another, even more fine-grained, classification layer to CATH's homologous superfamilies based on the function of a protein. However, even ColabFold [21] which provides important speed-ups to AF2 remains too slow to be applied to large protein databases with billions of sequences [6, 7]. Therefore, we investigate in this dissertation new ways to predict CATH classes from novel protein representations which are tailored towards capturing the CATH hierarchy (Chapter 5). Looking at the problem purely from a machine-learning perspective, we try to tackle a hierarchical multi-class classification problem that tries to predict thousands of highly imbalanced homologous superfamilies in CATH.

1.1.3. Protein-level Function: Subcellular Localization

Each prokaryotic or eukaryotic cell contains compartments, so-called *organelles* which allow it to provide different environmental conditions for performing highly specialized functions. For example, the genetic blueprint, the desoxyribonucleic acid (DNA), is stored and read in the nucleus but the final proteins are assembled in the ribosomes. Depending on a protein's role in the cellular machinery, it gets transported to its destination, a specific organelle or *subcellular localization*. In many cases this is conducted by "zip-codes" that are part of a protein sequence, so-called signal peptides, which get cleaved off after successful transport. This information offers an important starting point for unravelling the role of a protein in the complex cellular machinery which makes it a long-standing [22, 23], yet unsolved prediction problem in CB [24–27]. I will use subcellular localization as a proxy to assess to which extent the proposed representations capture protein function (Chapter 2 and [28]). Some proteins are located in multiple subcellular localizations and amongst those a subset adopts different functions depending on the organelle they are currently in (so-called moonlighting proteins [29]). However, today's available annotations are mostly limited to one subcellular localization per protein, presumably because those special cases are relatively rare. Hence, I treat subcellular localization prediction as a protein-level classification task, i.e., each protein gets classified into one of the 11 compartments defined elsewhere [30]. Again, the accuracy of different protein sequence representations used as input to the same ML device serves as a proxy for estimating the extent to which they make proteins machine-readable.

1.1.4. Protein-level Function: Gene Ontology

The *Gene Ontology* (GO) [31] makes protein function human- and machine-readable by mapping it to a structured and controlled vocabulary. Only such standardization efforts make experimental annotations of protein function comparable and enable unified computational processing. GO takes the fuzzy meaning of function into account by separating it into three hierarchies: On the molecular level, a the function of a protein is described by the Molecular Function Ontology (MFO) which is put into the larger context of the cellular process it is part of as well as the subcellular localization it functions in by the Biological Process Ontology (BPO) and the Cellular Component Ontology (CCO), respectively. Each of the ontologies is organized as directed acyclic graphs (DAG) with each node representing a functional annotation. To take account for the diverse spectrum of functions that a single protein can perform, each protein can be annotated to multiple functions of varying resolution, i.e., varying depth in the DAG, depending on the reported evidence. However, most proteins have incomplete GO annotations as there is no automated way to experimentally test for the vast spectrum of potential protein functions. This makes the prediction of a protein's GO term(s) a challenging yet relevant problem that was tackled from its introduction in the 2000s [32–34] to today [35–37]. From a ML perspective, the prediction of GO terms can be viewed as a multi-class and multi-label problem as each protein can be assigned to multiple functions. Further, each protein can have varying annotation specificity depending on the experimental setup which results in varying annotation depths within the hierarchy. This makes not only the prediction of the terms highly complicated but also the evaluation thereof as it depends on the specific use-case whether it is preferable to have more accurate but less specific GO term predictions or less accurate but potentially more fine-grained predictions. This complicates the comparison of GO prediction approaches which is why we followed the evaluation of the Critical Assessment of protein Function Annotation algorithms (CAFA) [38], an international collaboration that compares computational methods predicting GO terms (Chapter 3). By reproducing the conditions of this competition, we have a direct and unbiased comparison not only between our own

GO predictors based on the proposed representations but also to other methods that participated in the last round of CAFA.

1.1.5. Residue-level Function: Conservation

Protein sequence is much less conserved compared to protein structure [39]. Consequently, many proteins with vastly different sequences exist that still adopt the same 3D structure and hence perform the same or very similar function. Despite this variability, not all positions or residues in a protein are equally likely to change as some are central to maintain the function of a protein (so-called hot-spots) or structure, e.g., stabilizing long-range interactions, limiting their mutational variability [40]. This is also reflected by surface residues of a protein being on average less conserved compared to those at the core as surface residues are spatially less constrained. Correspondingly, highly conserved residues on a protein surface correlate with binding sites while conserved core motifs can be used to deduce evolutionary relationship of highly diverged proteins [41]. As there is no direct way to measure a residue's conservation experimentally, most approaches rely on proxies derived from, *inter alia*, the variability of a residue's position in MSAs [42], the distance in an evolutionary tree, its structure or all of those combined [43]. Looking at this problem purely from a ML perspective, the task is to classify each residue in a protein into variable or conserved residues and, depending on the definition, classes in between. Here, we used the conservation definition introduced by ConSurf [43] which classifies each residue in a protein into a class ranging from one (highly variable) to nine (highly conserved). The performance on this multi-class prediction problem was used to benchmark the predictive performance of various protein sequence representations (Chapter 4).

1.1.6. Residue-level Function: Variant effect

As protein structure and function is determined by the precise arrangement of AAs, any change thereof might break the delicate biophysical equilibrium that leads to this specific structure and allows to perform a certain function. Such single amino acid variants (SAVs) are involved in many diseases so understanding their impact plays a pivotal role in precision medicine [44]. On top, estimating the impact of SAVs is important when ranking mutants for efficient protein engineering. However, exploring the mutational landscape of a protein by experimental means is extremely expensive and time-consuming, making a large-scale application nearly impossible. The bottleneck here is the huge search space: every wild-type protein of length L can be mutated to L * 19 (19 possible other AAs) mutants. Additionally, the experimental assay would not only have to mirror in-vivo conditions but it would also have to capture all functions a protein performs. As a consequence, so-called Deep Mutational Scanning (DMS) experiments which mutate each AA in a protein to any other AA are only available for very few, relatively short proteins and usually measure only a single protein function [45]. Still, those data points are extremely valuable as they allow to benchmark to which extent SAV effect predictors can estimate the subtle differences between different SAVs. More commonly available are binary labels that distinguish *effect* from *neutral* mutants, irrespective of any nuances that different SAVs might have on protein function. Despite their more coarse-grained information, the relative abundance of samples with binary labels compared to DMS data, makes binary effect data a commonly used dataset for training SAV effect predictors. From a ML perspective, this is a residue-level classification task where each mutant is classified as whether or not it breaks a protein's function (neutral vs effect). Here, I compare the predictive performance of different protein sequence representations on capturing the disruptive effect of SAVs by using them as input for training binary classifiers (Chapter 4). We further evaluate the trained models on their ability to capture the nuances of different SAVs by evaluating on existing DMS data.

1.2. How to Represent Data? - The Traditional Way: Handcrafted Features

One of the first and most simple approaches to make protein sequences and their constituent parts, AAs, machine-readable is called *one-hot encoding*. This encoding assigns to each of the 20 standard AAs a numerical vector of 20 elements, consisting only of "0" and a single "1". The position of the single one differs between each of the 20 AAs, allowing for unambiguous mapping between the character representation of each AA to its corresponding numerical vector. When being applied to each residue in a protein sequence of length L, the human-readable, character representation is converted to a machine-readable, numerical matrix of shape L x 20. However, such a representation omits any bio-chemical or bio-physical knowledge about AAs. In the past, this lack of information was tackled by expanding the one-hot encoding vectors by *handcrafted* chemical properties for each AA encoded as numerical values. For example, nearly 100 different hydrophobicity scales were developed by domain experts over the last 60 years [46]. While those scales have a good correlation, they give slightly different results on the exact ranking of AAs as hydrophobicity can be measured in multiple ways leading to different experimental setups [47]. Generally speaking, handcrafted features are inherently bound by what we can realistically measure in experiments making it nearly impossible to unambiguously capture properties such as hydrophobicity that depend on their environment.

Similarly, hand-crafted features dominated how data is represented in other fields. For example, in computer vision (CV) the processing of images was dominated by handcrafted rules designed by domain experts such as the *Sobel filter* [48]. This filter creates a new representation by convolving two 3x3 filters over a given image, i.e., one filter is designed to detect horizontal edges while another filter is designed to detect vertical edges. While this is very similar to modern convolutional neural networks (CNNs) [49], the weights of the Sobel filter were chosen manually instead of trained from data, as handcrafted features are always bound by what we can express with algorithms at this time. Still, both approaches, the Sobel filter and handcrafted features for AAs, allowed machines to process certain types of real world observations, opening the door to automated processing. However, the main drawback of handcrafted features remains: They crucially miss the global picture of the underlying data, i.e., hydrophobicity scales and Sobel filters are limited to local properties, ignoring global context such as the complex interaction between multiple residues within the 3D structure of a protein or how to relate single edges to more complex concepts such as houses.

1.3. Leveraging Existing Data by Statistical Means Improves Representations.

With more and more data being available, researchers started to compare the (constantly increasing number of) available samples by statistical means. One of the key advantages of such approaches is that they allow to relate new samples against the background of existing, potentially annotated samples. For example, statistical analyses in the form of sequence alignments were crucial for the development of substitution matrices such as BLOcks SUbstitution Matrix (BLOSUM) [50] which estimates the log odds ratio of observing a certain SAV in a protein sequence. The values in these matrices were estimated from substitutions observed in local alignments between pairs of protein sequences. Despite being developed 30 years ago, BLOSUM is still being used by today's sequence search tools such as MMSeqs2 [51] to score the significance of alignments. A more sensitive but also much slower approach for searching for evolutionary related sequences is to translate an MSA to hidden-markov models (HMMs, [52]) and compare it to other HMMs as done by, e.g., HHBlits [53]. It is important to note that such statistical analysis did not supersede handcrafted features completely as the latter can be derived from the former. For example, some of the aforementioned hydrophobicity scales were derived from analysing a residue's average solvent accessible surface area from a pool of proteins with resolved 3D structures [54]. In a different line of work, combining the two approaches, sequence alignments and the comparison of protein 3D structures, allowed to unravel complex sequence-structure relationships such as the homology-derived secondary structure of proteins (HSSP)-curve [55, 56], which estimates the structural agreement of two proteins only from their sequence alignment even if the sequence identity is as low as 20%. None of these developments would have been possible without statistically relating single sequences against the background of large protein sequences database.

Summarizing the evolutionary information (EI) of related proteins in the form of MSAs gave rise to the arguably most influential way of making protein sequences machine-readable for machine-learning (ML) [13]. However, while the performance of nearly all modern predictors in CB, including AF2, rely crucially on EI summarized in MSAs the way those methods leverage this information differed over the years. Early approaches mostly relied on evolutionary profiles such as position-specific-scoring-matrices (PSSMs) [57] which capture the (log-)likelihood of observing a certain AA at a certain position in a protein. High values indicate that an AA occurs more often at a certain position than expected. This usually points towards conserved residues that are important to maintain a protein family's function. Similar to one-hot encoding, PSSMs are matrices of shape L x 20 with L being the length of a protein. Evolutionary profiles paved the way for high-quality prediction of protein features that are mostly encoded in the local sequence neighborhood, e.g., secondary structure [13, 15].

The problem of evolutionary profiles is that higher-order dependencies involving multiple residues, potentially well-separated in sequence- but close in structure-space are lost when converting a MSA to, e.g., PSSMs. This is problematic as residues close in 3D structure carry crucial information on co-evolving residues, i.e., a protein can only maintain its function upon mutation if another residue, usually spatially close to the first mutant, changes such that it accommodates or compensates for the first mutation [58]. Such co-evolving residues have been among the most informative signals for inferring a the structure of a protein as the entanglement in 3D directly affects the mutational landscape of a protein sequence. Given a sufficient number of evolutionary related proteins summarized as MSA, Pott's models [59] can estimate such evolutionary coupling scores (ECs) [60]. Paired with advances in ML, most importantly the introduction of CNNs, EC-based methods outperformed approaches building on evolutionary profiles in predicting protein 3D structures [61, 62]. As ECs provide coupling scores between all pairs of residues as well as all pairs of AA mutants, the resulting representation has a shape of L x L x 21 x 21 with L being again a protein's length, and 21 being the number of standard AAs plus a special gap character indicating deletions. So each cell in this 4D-tensor provides a score for observing a specific combination of AAs (21 x 21) for a specific pair of residues (L x L) in a given protein family. This way of representing proteins was also successfully applied to predict protein-protein-interactions (PPIs) which undergo similar evolutionary pressure as they have to maintain their interaction even upon mutations of the interaction interface [63]. However, in this case the co-evolving residues which compensate each others mutations are part of two different proteins. Further studies showed how this protein representation, i.e., ECs, can also be used to estimate the effect of SAVs [64] and the prediction of binding residues [65]. Despite this success, ECs required extremely large and sufficiently diverse MSAs to have enough statistical power to provide a signal clean enough to be useful [58]. Paired with the high computational demand required to compute ECs from MSAs, this limited the large-scale applicability of ECs as general-purpose representation of protein sequences.

ECs highlight how all statistical/evolutionary approaches are to varying degrees double-edged swords: their predictive power depends on the amount of diverse, yet evolutionary related sequences, one can find. This usually requires searching large protein sequence databases. Part of the success of methods in the recent CASP was that they expanded the search for related sequences to large metagenomic protein sequences databases with billions of proteins [6, 7]. While this increases performance, it also increases runtime for each new query. In turn, this also means that all EI-based approaches suffer from low representational power if no or only a few related proteins can be found in today's databases [66]. But most importantly, even if a sufficient number of related sequences can be collected for a protein of interest, certain use-cases such as the effect scoring of SAVs, benefit from protein-specific representations instead of family-averages over MSAs. One of the most recent and most prominent examples highlighting this problem was AF2, which provides extremely accurate protein 3D structure predictions but is mostly blind to SAVs in the input as the input MSA will be nearly identical for wild-type and mutant [67]. Speaking more generally, all statistical methods are limited to a) what we can easily measure, b) our current understanding of the underlying problem in general and c) to what extent we can formalize or extract this knowledge by computational means. As nature itself only requires single protein sequences for folding a protein to its functional 3D state, all information necessary to predict protein properties should be encoded in a single protein sequence [9]. This raises the question of whether there are alternative ways of processing proteins such that information is directly extracted only from single protein sequences, dropping the necessity to search for related proteins in databases.

1.4. Let Data Speak for Itself. - Learning Representations from Unlabeled Data

Together with the advent of big data, hardware solutions like graphical processing units (GPUs) or tensor processing units (TPUs) were either adapted (GPUs) or developed (TPUs) to allow for efficient processing of large amounts of data via neural networks. This development gave rise to a whole new field that sat off to find new ways to make real world observations machine-readable, i.e., *representation learning.* In contrast to previous approaches that relied on domain experts to design features or apply statistical means, these new representations are learnt directly from raw data without any human intervention. Roughly a decade ago, the field of CV was the first to replace handcrafted features by fully *end-to-end* learnt representations, i.e., raw pixels were used as input to let the network decide itself how to optimally weight each feature when making a certain prediction [68]. This development allowed predictions at an unprecedented level of accuracy without the necessity to manually design filter kernels such as the Sobel filter. Instead of manually designing new filters, ML experts now rather try to understand the inner workings of the neural networks they had trained to improve the performance and efficiency of the models' architecture [69, 70]. Such end-to-end trained solutions had not been possible before as a variety of developments had to synergize with each other: a) large amounts of labeled data had to be collected (e.g., imagenet for CV [68]), b) the necessary hardware had to be adapted (GPUs) or developed (TPUs) and c) new neural network architectures had to be developed to either fit the underlying data type (e.g., grid-like architecture of CNNs for images) or to provide enough flexibility to extract relevant information irrespective of the domain/problem, e.g., Transformers [1]. Only the clever combination of those allowed recent solutions to improve over handcrafted/statistical features that were optimized over decades. Of course, this is not a purely sequential development where new representations always supersede existing approaches but there are also solutions that combine existing representations and novel ways to extract information from it. For example, the highly accurate protein 3D structure prediction of AF2 still crucially relies on explicit EI in the form of MSAs as input. However, instead of manually extracting features from it using handcrafted or statistical approaches, AF2 was trained end-to-end, i.e., a special type of Transformer learnt how to weight and combine the different sequences in the MSA input. An overview of the different ways to make protein sequences machine-readable is given in Figure 1.2.

1.4.1. How to Leverage Unlabeled Data?

Despite this success, most approaches still required labeled samples which posed a problem for scaling this approach up and for transferring the concept to domains with less labeled data compared to CV. This led to the emergence of a new sub-field within representation learning called *self-supervised learning* that tapped into the vast gold mine of unlabeled data. Arguably, this trend mainly emerged from NLP where labeled data is more scarce compared to CV, which hindered the large-scale application of representation learning to textual data. Instead of relying on labeled data, self-supervised learning solely relies on the signal inherent to sequential data itself thereby removing any reliance on labeled data. One of the first approaches to harness the sequential structure of textual data was word2vec [71] which introduced two basic concepts: a) similar to a cloze test, train a feed-forward neural network (FNN) on re-constructing the center word given all other words within a certain window-size (CBOW) or b) vice-versa train a FNN on reconstructing the surrounding words given only a center word within a window of words (skip-gram).

While the predictions themselves were only of limited use, the network had to learn certain patterns within the sequential data to do well in the tasks. Based on the context a word appears in and its co-occurrence with other words, the network learnt commonalities between words, and to a certain extent, semantic and syntactic similarity. The knowledge acquired during this so-called *pre-training* phase could later be transferred to any other application by removing the final classification layer and instead extracting

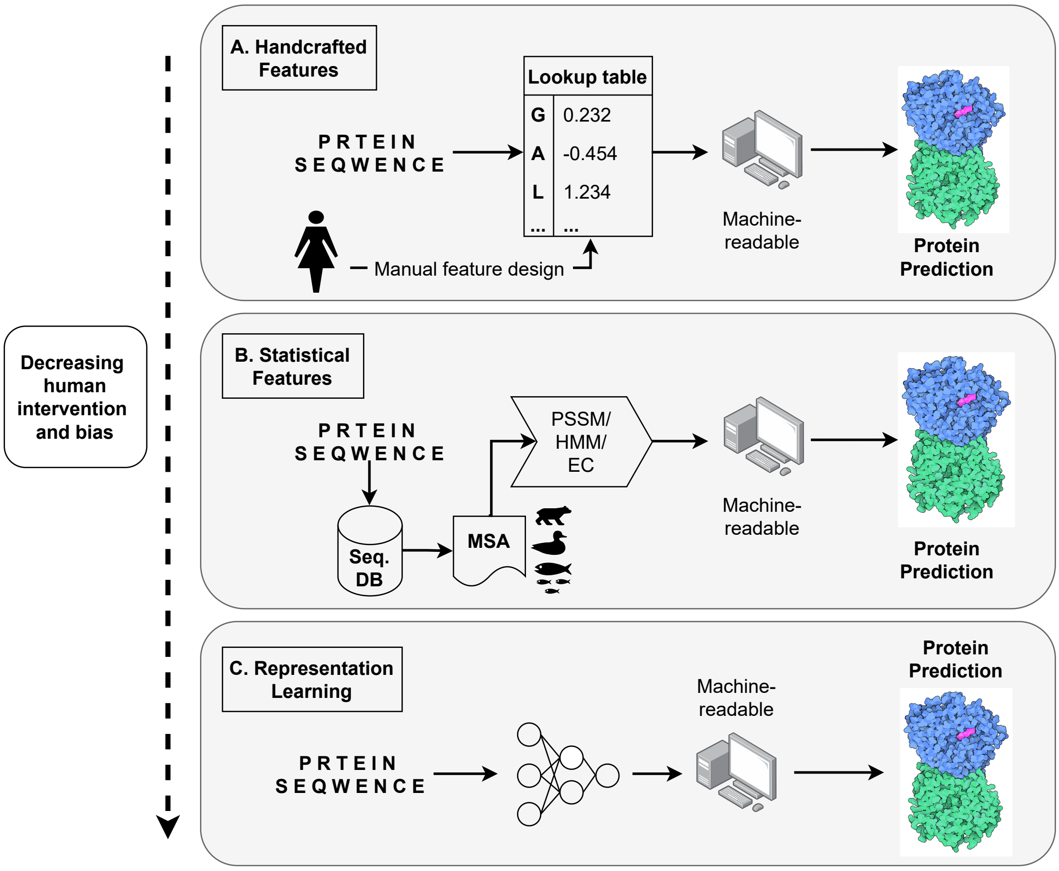

Figure 1.2.: Overview of different ways to make protein sequences machine-readable. Panel A depicts how domain experts manually create features. Those *handcrafted features* such as various hydrophobicity scales can be used to encode amino acid letters as numerical values capturing certain protein properties. This gets extended by Panel B with domain experts creating algorithms to search through large sequence databases to find evolutionary related proteins by *statistical means.* Those related sequences are summarized in a multiple-sequence alignment (MSA) from which various numerical representation can be computed, i.e., position-specific scoring matrices (PSSMs), hidden-markov-models (HMMs) or evolutionary coupling scores (ECs). In this work, I introduced and evaluated novel ways of representing protein sequences as numerical vectors by learning representations directly from unlabeled sequence data (Panel C). While there is a certain chronological order from established (Panel A) to novel (Panel C) representations, this is no strict trend as statistical means were deployed to derive new handcrafted features. However, more importantly there is a decrease in human intervention from handcrafted features to learnt representations which reduces the risk of introducing human bias.

the intermediate vector representation output by the hidden layer(s) for a given input (called *embeddings*). Those embeddings can now serve as input for any other application centered around automated processing of textual data. For this reason, despite being more complex and much larger, today's equivalents of word2vec are called *foundation models* as they form the bedrock for many modern predictors that build on top of embeddings. On a more general level, training a model on one task and transferring the thereby acquired knowledge to another task is called *transfer learning.* As word2vec-like approaches do not rely on any domain-specific concepts or assumptions such as grammar but solely on the signal inherent to the structure of sequential data, they can readily be applied to other domains working with such types of data. The sequential nature of protein sequences lend itself to be applied to methods such as word2vec as showcased by ProtVec [72]. The authors of ProtVec trained a word2vec architecture on around 500k protein sequences taken from SwissProt [73]. However, in contrast to most languages, protein sequences are one long consecutive string of characters without clear word boundaries that are required by word2vec-like approaches. The ProtVec authors bypassed this problem by simply splitting each protein into 3-mers (three consecutive AAs) and training the model on reconstructing the surrounding 3-mers given the center 3-mer in windows of size 25. While the final output of ProtVec is only of limited use, the model is forced to learn co-occurring patterns that relate to higher-level concepts ranging from bio-chemical and bio-physical properties of AAs over structural and functional aspects to the evolutionary relationship of proteins [72]. In the case of ProtVec, this information was extracted by translating each 3-mer in a protein sequences to embeddings of length 100. Those vectors were used to classify, *inter alia*, protein families and disordered proteins.

The basic idea of learning general concepts directly from the wealth of information arising from gigantic amounts of unlabeled data and later transferring the acquired knowledge to other tasks with a limited set of labeled data might appear straightforward today but the techniques arising from it did not only revolutionize NLP but also many other fields such as CB. The importance of this revolution probably is best reflected by the fact that essentially all applications working with text, e.g., automated translations [74], speech-to-text conversion, question-answering [75] or document retrieval [76], are powered today by models invented in the course of this revolution.

Still, all word2vec-like approaches [77, 78] have one major drawback: They inevitably return the same embedding for the same word, irrespective of the context it appears in which is why they are considered to be *un-contextualized*, i.e., they can not capture the ambiguity inherent to many words. For example, word2vec-like approaches always provide the same embedding for the word *tiger*, irrespective of whether the context clearly refers to a toy made of paper, or a threatening animal. Consequently, ProtVec can not render residues or 3-mers depend on neighboring residues making it impossible to capture the complexity of proteins where the role of each residue depends on its surrounding residues in sequence and structure space.

1.4.2. Context is Key - The Rise of Language Models

Language models (LMs) were introduced to overcome the limitations of uncontextualized approaches by rendering embeddings dependent on their context. Among the first LMs was a model called Embeddings from Language Models (ELMo [79]) which trained a stack of long-short-term-memory cells (LSTMs [80]) on predicting the next word in a sentence given all previous words of the same sentence (*auto-regressive language modeling*, ALM). LSTMs are a specific type of recurrent neural networks (RNNs) that allow to process sequential data by applying a modified FNN not only to new input but also to its own previous output. This message-passing allows the network to "memorize" previous input and allows to process input of variable length as opposed to chunking sentences into pre-defined windows of words as done by word2vec. Additionally, the conditioning introduced by ALM enables the network to produce different embeddings for the same word, i.e., to created *contextualized* embeddings. The sequential processing of LSTMs also introduces an implicit positional bias allowing the network to learn concepts at different resolutions, i.e., the network itself learns to weight local structures over global concepts and how to best relate them. Similar to word2vec, repeating this pre-training task on millions of sentences, forces the model to learn certain commonalities and statistical patterns solely from the sequential signal in the input, i.e., without any labeled data. After this pre-training, transfer learning is again applied by generating embeddings from the hidden layers of the LSTM for a given input. This way, the knowledge acquired during pre-training can be transferred to other tasks, potentially with little labeled data. It was shown that the combination of self-supervision and transfer learning particularly helps problems with limited labeled data [75]. The authors assume that embeddings often contain enough information about a domain so that simple regressions suffice to reach high performance while avoiding overfitting thanks to the strong regularization introduced by the limited number of free parameters of the regression [75]. The

ELMo authors showcased the benefit of their proposed approach by comparing against their uncontextualized counterparts on a variety of different tasks including question answering and sentiment analysis.

Thanks to the sequential nature of both, natural language and protein sequences, we could transfer the concept introduced by ELMo directly to protein sequences. Instead of splitting proteins into 3-mers as done by word2vec, we treated each single AA as a word and each protein sequence as a sentence. This had the advantage that we could extract embeddings for each individual residue in a protein. Consequently, the optimization task became predicting the next AA given all previous AAs in a protein sequence. This allowed the resulting model, dubbed SeqVec [81], to leverage the information stored in exponentially growing but unlabeled protein sequence databases (Chapter 2). For SeqVec, we used 50M unlabeled protein sequences in UniRef50 [82], a version of UniProt [83] clustered at 50% sequence identity. We evaluated the knowledge acquired during pre-training, by generating embeddings from the hidden layers of SeqVec and using those representations as input for various established CB tasks such as secondary structure prediction or subcellular localization prediction. Each protein sequence of length L results in a embedding of shape L x 1024 (with 1024 being a hyperparameter defined by the user before pre-training), so each residue in a protein is represented by a 1024-dimensional vector. We also showed that simple averaging (average pooling) over the length-dimension of protein embeddings allows to project each protein to a fixed-size vector (1024-d), irrespective of the length of a protein.

However, ALM as introduced by ELMo cannot capture true bi-directional context, i.e., predicting the next word given all previous words in a sentence usually requires either processing left-to-right or right-to-left but no mix thereof as the model could otherwise cheat rendering the optimization task trivial. ELMo tries to compensate for this by independently processing the same sentence with two different stacks of LSTMs once from left-to-right and once from right-to-left (see Figure 1.3 Panel 1). Some parameters are shared between the two networks, i.e., the initial token representation and the final softmax classification, but besides this, the two networks maintain an independent set of parameters that are jointly optimized by summing the losses of the two directionalities. While this produces two sets of embeddings, each capturing context in one direction, it does not render the embeddings themself truly bi-directional as each of them only has access to one direction. This makes it impossible to learn concepts that rely on gathering information from both directions. Such a problem might be less severe for

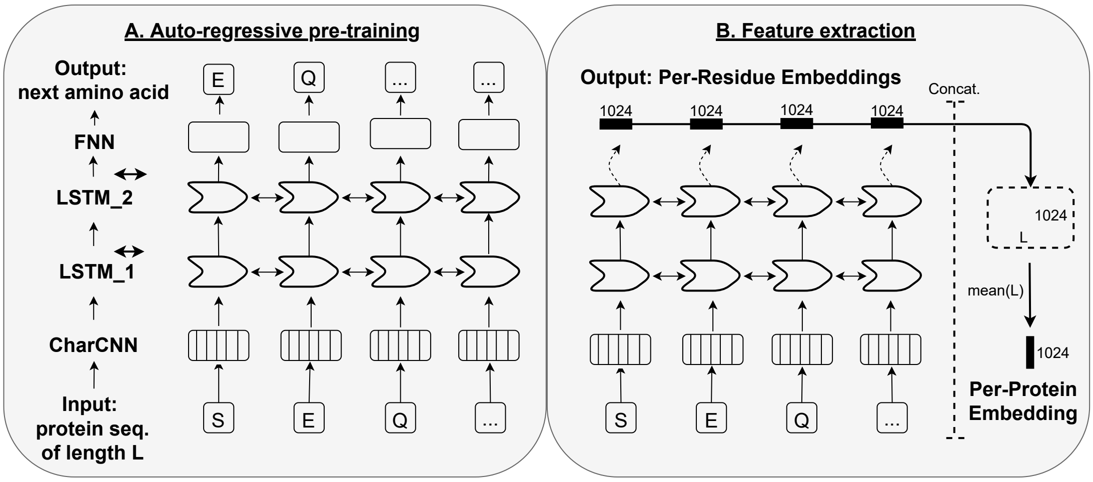

Figure 1.3.: Representation learning can be split into two major parts: the self-supervised pre-training which requires only unlabeled sequential data and its subsequent deployment for e.g. feature extraction. Panel A shows the auto-regressive pre-training used for optimizing ELMo/SeqVec (Chapter 2): A charCNN is used to encode AA letters as numerical values and two stacked bi-directional LSTMs are used to gather information from previous tokens. A final FNN predicts the next AA from all previous AAs in a protein sequence (auto-regressive pre-training). Only after this pre-training has finished, the acquired knowledge can be transferred to other tasks via feature extraction (Panel B). Towards this end, the final classification layer used to predict the next AA, i.e., the FNN, gets removed and instead the hidden states of the LSTM layer(s) are extracted. The resulting numerical representations for each AA (*per-residue embeddings*) can be concatenated and by averaging over the length-dimension (mean-pooling), fixed-size representations can be derived irrespective a protein's length (*per-protein embeddings*).

natural language which is designed to be readable either left-to-right or right-to-left, depending on the language. However, capturing context in both directions is essential to capture the complexity arising from protein sequences folding to 3D macromolecules.

1.4.3. The Transformer Revolution

While LSTMs are able to capture the context of a word to a certain extent, they struggle to capture long-range dependencies as the history of a token has to be remembered over hundreds or thousands of steps. Similar to the children's game "whisper game", the more intermediate stages or tokens, the harder it becomes to maintain the original message. In CB, long-range interactions between residue-pairs are of particular importance as protein sequences naturally adopt a 3D-fold to perform their function. During folding, AAs far apart in sequence space (hundreds of residues) can come into close structural proximity making it essential for CB to have a network that can handle such dependencies easily.

This dual nature of proteins perfectly suited a novel branch of neural network architectures, *Transformer* [1] models. At their core, Transformers rely on the *attention mechanism* [84] which was originally proposed to perform translation tasks. However, in this first introduction, attention was used to connect the output of an encoder LSTM that reads a source language to the input of a decoder LSTM that generates a translation to a target language. Only later, the authors of the paper "Attention is all you need" [1] showcased that removing the LSTMs and instead solely stacking multiple attention-layers into an encoder-decoder architecture improves translation quality. However, this original Transformer implementation did not benefit from unlabeled data but instead learnt to translate from one language to another using labeled data. Still, this work arguably was the starting point for a ML-revolution as it showcased the Transformer's flexibility regarding the expected input. As those models only require a set of input "tokens" with variable length, the Transformer allowed for wide-spread application far beyond processing of textual data, ranging from processing images in CV [85], over geometric data [86], to StarCraft units [87]. From a high level perspective, the attention mechanism relates all pairs of tokens in a set of inputs by computing scores for each pair of tokens. In this context, tokens are not limited to words or characters in a sentence but can refer to, *inter alia*, (crops of) images, 3D coordinates of point clouds, or (sets of) StarCraft units. Irrespective of the domain or input, the main goal of the attention mechanism remains the same: For a given set of input tokens, learn to highlight the salient information/feature that is required to solve a specific task. For example, during translation this mechanism learns to highlight which words in the source language to attend to while generating each word in the target language. To achieve this, the general attention mechanism relies on three different abstract components: a set of query vectors q, a set of key vectors k, and a set of value vectors v. From those, a weighted-sum for each input token over all other input tokens is computed by "searching" each query (e.g., a single word in the input sentence) against a "database" of keys (e.g., all words in the input sentence) and using the resulting scores/weights to compute the weighted-sum over all values (all words in a sentence). This results in a new vector for each input token that now does not only hold information on the token itself but also all other tokens in the input. To allow the network to learn this weighting during training, q, k, and v vectors are generated by multiplying the initial token vectors (e.g., word vectors) with trainable weight matrices, W_Q, W_K, and W_V, respectively. For the de-facto standard attention, the *scaled dot-product attention* [1], the dot product between each query q and all keys k is being computed. The resulting vectors are normalized by dividing by the square root of the vector dimension of the keys to avoid exploding/vanishing gradients. The soft-max function is applied to the normalized vectors to derive a weight for a given q with respect to all keys k. Those weights are used to compute a weighted sum over all values v to create a new vector for query q that now contains information on all other input tokens. Using trainable weight matrices (W_Q, W_K, and W_V) in this process allows the network to learn the weighting of input tokens that is optimal for a given task. To allow for parallel processing, the q, v, and k vectors are usually each concatenated to matrices Q, K, and V. Depending on how queries, keys, and values are defined, one can either compare all pairs of input tokens within one set (*self-attention*) or between two sets (*cross-attention*). The latter lends itself for automated translation by basing keys and values on a source language while basing queries on a target language. In the original Transformer paper, both attention types were mixed by using an encoder with self-attention to generate embeddings for a sentence in the source language which are then interlaced with embeddings from the target language using cross-attention. Later architectures relied mostly on either the encoder [2] or the decoder [88]. Cross-attention can also be used to mix different modalities, e.g., using image embeddings as queries and basing keys and values on associated text (e.g., image caption) allows interlacing those two modalities in a single, shared embedding space [89]. In practice, multiple attention layers are trained in parallel together with how to combine their information (*multi-head attention*) to allow the network to focus on different aspects of the input. Additionally, multiple layers of attentions are usually stacked to give the network sufficient power to capture the complexity of the input by allowing to detect salient features on multiple levels of granularity.

Besides this unprecedented flexibility on the expected input format, Transformers come with various other advantages that made them indispensable for many fields including representation learning: a) Compared to LSTMs which process their input sequentially, Transformer's self-attention processes input data inherently parallel, speeding up especially processing of long sequences; b) where LSTMs had to remember information over many steps to detect long-range interactions, Transformers require only a single computational step to accomplish this (one query-key comparison) simplifying learning of such concepts; c) while LSTM-based language models could not generate truly bi-directional embeddings, Transformer's self-attention computes weighted-sums over all input tokens, irrespective of their position. However, those advantages are not for free: The pairwise comparison of all input tokens results in quadratic memory consumption for standard Transformers. This does not only affect the size of the Transformer or the number of samples that are processed simultaneously (batch size), but most importantly, this means that the available GPU/TPU memory sets an upper limit for the maximum sequence length to be processed. Also, the flexibility of a Transformer regarding its input introduces new problems for processing sequential data: As self-attention is a set-operation and thus order independent, an explicit positional encoding has to be added to a Transformer's input. Otherwise, the Transformer could not relate words in a sentence to their position, as it would only see a random collection of words instead of a grammatically correct sentence. Towards this end, the authors of "Attention is all you need" proposed to sum the input with a sine and cosine function of different frequencies.

1.4.4. The Transformer Landscape. - A Brief Overview

While the original Transformer still relied on an encoder-decoder architecture and used labeled data for training, it paved the way for a variety of attention-based models that use either the Transformer's encoder or decoder or both to learn novel representations directly from unlabeled data. This raised the question of how to best leverage unlabeled data with those novel architectures. The authors of OpenAI's *GPT* (Generative Pre-trained Transformer) [88] showed that the generative nature of the Transformer's decoder designed for generating translated text, allowed for straightforward application of ALM. Simply adding a causal masks to the input prevented the decoder from accessing information of future tokens and allowed pre-training of Transformer decoders on unlabeled data. While this approach improved over LSTM-based LMs such as ELMo, it did not solve the problem of generating truly bi-directional embeddings. Instead, the authors of *BERT* (Bidirectional Encoder Representations from Transformers) [2] proposed a solution that solely relied on self-attention by stacking multiple Transformer encoder layers. However, this idea clashed with the idea of ALM: How do we avoid that the Transformer accesses information on the future tokens that we want to predict while still providing bi-directional access? The authors of BERT proposed *Masked Language Modeling* (MLM), which is inspired by the cloze test [90], to solve this problem. During MLM, some fraction (usually 15%) of the input tokens is randomly corrupted, and the model is trained on re-constructing these corrupted tokens from the remaining, non-corrupted context. This modification allowed the authors of BERT to outperform GPT on a variety of NLP tasks, highlighting the benefit of bi-directional context. However, it has to be noted that GPT trained via ALM has an advantage over MLM models with respect to sample efficiency: While the MLM loss is only computed over the subset of corrupted tokens, ALM-based methods compute their loss over all input tokens making them more efficient during optimization.

Since the introduction of the MLM pre-training and architectures consisting solely of the attention mechanism, modifications and improvements to those concepts are proposed nearly on a monthly basis highlighting the transformative impact of those techniques. BERT was the first bi-directional model in NLP which tried to reconstruct corrupted tokens and is arguably the de-facto standard for transfer learning in NLP. While BERT trained multiple attention layers without sharing parameters, *Albert* [91] reduced BERT's complexity by hard parameter sharing between its attention layers, i.e., a single attention layer is applied multiple times to its own output to save parameters. To still benefit from the presumably advantage of larger architectures, the authors of Albert proposed to increase the number of attention heads, thereby giving the single attention layer more options to zoom into different features. *Electra* [92] took a different starting point and aimed to improve the sample-efficiency of BERT. Electra tries to improve over this strategy by training two different networks, a generator and a discriminator. The generator (BERT) still reconstructs masked tokens, potentially leading to plausible alternatives, while the discriminator (Electra) gets optimized to detect which tokens were masked. This way, the loss is computed over all tokens instead of only the relatively small subset of corrupted tokens. While BERT, Albert, and Electra only relied on the Transformer's encoder, the *Text-to-Text Transfer Transformer* (T5) [93] uses the original encoder-decoder Transformer architecture proposed for sequence translation. In order to use this architecture for self-supervised learning, the authors of T5 combined ideas of ALM and MLM via *teacher forcing*, i.e., input and targets were fed to the model with inputs being corrupted protein sequences fed to the encoder and targets being identical to inputs but shifted to the right and fed to the decoder. Additionally, the authors of T5 proposed to corrupt not only a single token but instead multiple consecutive tokens (span-based corruption). Further, they proposed to replace the explicit positional encoding by an trainable alternative that enables the model to learn the relative offset between tokens. The rational behind the latter was that learnt positional encoding allowed the model to potentially generalize to sequences longer than the ones seen during training. Taken together, these changes allowed T5 to reach state-of-the-art results in multiple NLP benchmarks. However, by relying on an encoder-decoder architecture, T5 is even more affected by the quadratic memory consumption, as not only all pairs of tokens input to the encoder need to be related but also all tokens input to the decoder. This sets an effective upper limit to the model size and to the maximum sequence length that can be processed by the model. In contrast to this, *TransformerXL* [94] introduced a memory that allows it to process sequences of arbitrary length. While the model still needs to cut sequences into fragments, it allows for flow of information between fragments by re-using hidden states of previous fragments. However, TransformerXL captures only uni-directional context within one fragment as well as between fragments (auto-regressive). In contrast to this, *XLNet* [95], which uses a similar memory mechanism to process sequences of arbitrary length, allows to gather bidirectional context within one memory fragment.

We investigated how various Transformer models, i.e., BERT, Albert, Electra, T5, TransformerXL and XLNet, can be trained on proteins by equating words in a sentence with AAs in a protein sequence [28]. This way, we can readily apply ALM, MLM or T5's span-based reconstruction for pre-training on large, unlabeled protein sequence databases such as the metagenomic database BFD with 2B protein sequences [6, 7]. With the resulting models, dubbed ProtTrans, we were together with a team at Facebook Research [96] among the first to show that embeddings generated from single protein sequences can outperform established state-of-the-art methods relying on EI summarized as MSAs. This does not only remove the computationally expensive database search needed to generate MSAs but also allows for protein-specific predictions instead of family-averages as inherently produced by EI-based methods. We showcased the advantages of the proposed solution by training a variant-effect scoring prediction without alignments (Chapter 4 - *VESPA*) and by developing a fast homology-detection beyond sequence similarity (Chapter 5 - *ProtTucker*).

1.4.5. Zero-shot learning and Homology-based Inference

Together with the rise of contextualized language models which allow to project sequential data to meaningful vector representations (embeddings), *zero-shot learning* became increasingly popular. In contrast to other learning concepts, zero-shot learning aims at classifying samples from classes that were never observed during training or, more broadly speaking, to evaluate a model on a task it was not specifically trained on. Usually, some sort of encoder is trained to project data samples in an embedding space that reflects similarity between samples. In a next step, zero-shot learning is applied by using the distance between embeddings to transfer annotations from a set of labeled data points to a set of unlabeled data points. This way of annotation transfer allows to "predict" class labels that were never seen during training by solely relying on the encoder's capability to capture similarity. This concept is very similar to traditional homology-based inference (HBI) in CB, where annotations are transferred from a set of labeled proteins to unlabeled query proteins based on sequence similarity. In this dissertation, I showed that *per-protein* embeddings derived from protein language models (pLMs) still carry enough information to outperform established homology-based inference (HBI) for annotation transfer (Chapter 3 - goPredSim). Instead of sequence similarity as used in HBI, embedding-based annotation transfer (EAT) transfers annotations from a set of labeled lookup proteins to a set of unlabeled query proteins based on smallest Euclidean distance in embedding space.

1.4.6. Learning Representations from Contrasting Samples.

Along the same lines of representation learning which aims at learning novel representations of data, the sub-field of contrastive representation learning or short *contrastive learning* also aims at learning new embedding spaces [97]. However, where representation learning usually learns new representations from data itself without making any assumptions, contrastive learning usually depends on some notion of similarity between data points. During training, contrastive learning uses this notion of similarity to shape the new embedding space towards pushing apart samples that are considered dissimilar while pulling together samples considered similar. One of the most common ways to achieve this, is to pass triplets of samples to the network: an anchor sample, a sample that is considered to be similar to the anchor (positive), and *vice versa* a sample that is considered dissimilar to the given anchor (negative). During training, the network reduces the embedding distance between anchor and positive while increasing the anchor-negative distance. From a high-level perspective, this resembles learning a clustering in high-dimensional embedding space. Only recently, solutions were proposed that make use of multiple positives and negatives for a given anchor [98]. This strategy improves optimization by avoiding trivial triplets that are already classified correctly. Similarly, hard negative mining aims at sampling negatives not at random but according to some notion of difficulty [99]. Both approaches, hard negative mining [99] as well as sampling multiple negatives/positives [98], were shown to be crucial for performance as they avoid sampling trivial triplets. Many applications of contrastive learning such as signature verification [97] or face recognition [99] originated from CV, but the generality of the idea to bias a newly learnt embedding space towards capturing selected properties makes contrastive learning today also a popular choice in other fields. While there were also approaches that tried to combine self-supervised- and contrastive learning [100, 101], the resulting representations did not outperform embeddings from Transformers trained via ALM or MLM when being solely applied on unlabeled data. However, contrastive learning offers an interesting alternative for supervised use-cases that are hard to cover by established methods such as classification [102]. For example, for face recognition, not only the space of possible outputs explodes quickly for databases with millions of different persons but there are also only very few samples (images of faces) available for each class/person. In such cases, contrastive learning offers an alternative as it rather learns to sort the underlying data and how to access it instead of providing class predictions.

In this dissertation, I applied contrastive learning on top of embeddings derived from a pLM trained previously by us (ProtT5 [28]) to capture information from CATH - Chapter 5). The resulting model, dubbed ProtTucker, was optimized on similarity notion introduced by CATH, i.e., proteins with different CATH annotations were pushed apart while proteins with the same CATH annotation were pulled together. By applying EAT in the newly learnt embedding space, I was able to provide a fast and reliable prediction method for CATH that outperformed even sophisticated HBI-methods. Especially, for proteins with little sequence similarity, our proposed solutions improved over existing approaches.

1.5. Outline of This Work

This dissertation aims at introducing, applying and evaluating various techniques from representation learning to protein sequences. While existing solutions usually rely on features designed by domain experts, representation learning aims at learning representations/embeddings directly from large, potentially unlabeled data without any human intervention. Towards this end, I will use the LSTM-based language model ELMo in Chapter 2 to showcase how NLP concepts can be adapted to protein sequences [81]. By treating single AAs as words and protein sequences as sentences, I could show that our model, dubbed *SeqVec*, learnt meaningful representations for protein sequences using solely unlabeled data. Despite relying only on single protein sequences, our method *go-PredSim* [37] exemplifies the predictive power of the proposed representations as shown in Chapter 3, where we used embeddings derived from SeqVec to predict the function of a protein as defined by Gene Ontology (GO) terms. Similar to homology-based inference (HBI), which transfers annotations from a labeled lookup set to an unlabeled query set, our approach predicts GO terms via similarity search. However, instead of defining similarity as sequence similarity, we use vector distance in embedding space for embedding-based annotation transfer (EAT). In a work not described in more detail in this thesis [28], we improved over SeqVec by training and evaluating different Transformer types on large, unlabeled sets of protein sequences [28]. The predictive power as well as the general applicability of the resulting ProtTrans embeddings is exemplified in Chapter 4, where we use embeddings from our best performing ProtTrans model (ProtT5) for *Variant Effect Score Prediction without Alignments* (VESPA, [103]) [103], i.e., we show how embeddings can be used to predict the effect of SAVs. Finally, we show how contrastive learning can refine embeddings from ProtT5 towards capturing the hierarchical classification of protein 3D structures in Class, Architecture, Topology and Homologous superfamily as defined by CATH (*ProtTucker*,Chapter 5, [104]). Finally, a brief summary of all results presented in this dissertation is given in Chapter 6.

2. Modeling the Language of Life - SeqVec

2.1. Preface

Until recently, most state-of-the-art approaches used evolutionary information (EI) condensed in multiple sequence alignments (MSAs) as input to machine learning (ML) devices to predict functional or structural aspects of proteins [13, 15]. While those approaches improved over time by employing more sophisticated machine learning algorithms and the way they leveraged the sequences in the MSA [3, 105], some major limitations inherent to EI remained. Besides the compute time which can be a limiting factor for some applications, EI will remain less powerful for small families, e.g., for proteins from the Dark Proteome [106] or disordered proteins which are hard to align. Additionally, EI will remain less informative for some applications that require protein-specific instead of family-averaged predictions, e.g., single amino acid variant (SAV) effect prediction.

We introduced a novel way to represent single protein sequences as continuous vectors (embeddings) by using the language model ELMo [79] taken from natural language processing. By modeling protein sequences, ELMo effectively captured the biophysical properties of the language of life from unlabeled big data (UniRef50 [82]). This provides a single-sequence based alternative over EI to make protein sequence machine-readable. We refer to these new embeddings as *SeqVec* (Sequence-to-Vector) and demonstrate their effectiveness by training shallow convolutional neural networks (CNN) for two different tasks. At the per-residue level, secondary structure (Q3=79%±1, Q8=68%±1) and regions with intrinsic disorder (Matthew's correlation coefficient (MCC) =0.59±0.03) were predicted significantly better than through one-hot encoding or through word2vec-like approaches [72] which do not capture sequence context in their embeddings. At the per-protein level, subcellular localization was predicted in ten classes (Q10=68%±1) and membrane-bound were distinguished from water-soluble proteins (Q2=87%±1). Although SeqVec embeddings generated the best predictions from single sequences, no solution improved over the best existing method using EI (secondary structure [16]: Q3=85%, Q8=74%; disorder [16]: MCC=0.66; subcellular localization [30]: Q10=78%; Q2:Q2=92%). Nevertheless, our approach improved over some popular methods using EI and for some proteins even beat the best. Thus, the proposed embeddings prove to condense the underlying principles of protein sequences. Besides this proof-of-principle, the important novelty is speed: where HHblits [53] needed on average about two minutes to gather EI for a target protein, SeqVec created embeddings on average in 0.03s. As this speed-up is independent of the size of growing sequence databases, SeqVec provides a highly scalable approach for the analysis of big data in proteomics, i.e., microbiome or metaproteome analysis.

Thanks to Konstantin Schütze (TUM), the method is not only available as GitHub repository but also as a pip-installable package: https://github.com/Rostlab/SeqVec . Additionally, embeddings and predictions thereof can be generated via the *bio_embeddings* package [107].

Author contribution: I contributed to the conceptualisation, provided the language model training data, performed all evaluations, created the original GitHub repository, wrote the initial draft as well as the final, revised version of the manuscript. Ahmed Elnaggar contributed to the conceptualisation and trained the language model. Christian Dallago implemented the web server. All authors drafted the manuscript.

2.2. Journal Article: Heinzinger, Elnaggar *et al.*, BMC Bioinformatics (2019)

Reference: Heinzinger, M., Elnaggar, A., Wang, Y., Dallago, C., Nechaev, D., et al. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC bioinformatics*, 20(1):1–17, 2019

Heinzinger *et al. BMC Bioinformatics* (2019) 20:723
https://doi.org/10.1186/s12859-019-3220-8

BMC Bioinformatics

RESEARCH ARTICLE Open Access

Modeling aspects of the language of life through transfer-learning protein sequences

Michael Heinzinger[1,2*†], Ahmed Elnaggar[1,2†], Yu Wang[3], Christian Dallago[1,2], Dmitrii Nechaev[1,2], Florian Matthes[4] and Burkhard Rost[1,5,6,7]

Abstract

Background: Predicting protein function and structure from sequence is one important challenge for computational biology. For 26 years, most state-of-the-art approaches combined machine learning and evolutionary information. However, for some applications retrieving related proteins is becoming too time-consuming. Additionally, evolutionary information is less powerful for small families, e.g. for proteins from the *Dark Proteome*. Both these problems are addressed by the new methodology introduced here.

Results: We introduced a novel way to represent protein sequences as continuous vectors (*embeddings*) by using the language model ELMo taken from natural language processing. By modeling protein sequences, ELMo effectively captured the biophysical properties of the language of life from unlabeled big data (UniRef50). We refer to these new embeddings as *SeqVec* (*Seq*uence-to-*Vec*tor) and demonstrate their effectiveness by training simple neural networks for two different tasks. At the per-residue level, secondary structure (Q3 = 79% ± 1, Q8 = 68% ± 1) and regions with intrinsic disorder (MCC = 0.59 ± 0.03) were predicted significantly better than through one-hot encoding or through Word2vec-like approaches. At the per-protein level, subcellular localization was predicted in ten classes (Q10 = 68% ± 1) and membrane-bound were distinguished from water-soluble proteins (Q2 = 87% ± 1). Although *SeqVec* embeddings generated the best predictions from single sequences, no solution improved over the best existing method using evolutionary information. Nevertheless, our approach improved over some popular methods using evolutionary information and for some proteins even did beat the best. Thus, they prove to condense the underlying principles of protein sequences. Overall, the important novelty is speed: where the lightning-fast *HHblits* needed on average about two minutes to generate the evolutionary information for a target protein, *SeqVec* created embeddings on average in 0.03 s. As this speed-up is independent of the size of growing sequence databases, *SeqVec* provides a highly scalable approach for the analysis of big data in proteomics, i.e. microbiome or metaproteome analysis.

Conclusion: Transfer-learning succeeded to extract information from unlabeled sequence databases relevant for various protein prediction tasks. SeqVec modeled the language of life, namely the principles underlying protein sequences better than any features suggested by textbooks and prediction methods. The exception is evolutionary information, however, that information is not available on the level of a single sequence.

Keywords: Machine Learning, Language Modeling, Sequence Embedding, Secondary structure prediction, Localization prediction, Transfer Learning, Deep Learning

* Correspondence: mheinzinger@rostlab.org; assistant@rostlab.org
†Michael Heinzinger and Ahmed Elnaggar contributed equally to this work.
[1]Department of Informatics, Bioinformatics & Computational Biology - i12, TUM (Technical University of Munich), Boltzmannstr. 3, 85748 Garching/Munich, Germany
[2]TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany
Full list of author information is available at the end of the article

Background

The combination of evolutionary information (from Multiple Sequence Alignments – MSA) and Machine Learning/Artificial Intelligence (standard feed-forward artificial neural networks – ANN) completely changed protein secondary structure prediction [1–3]. The concept was quickly taken up [4–8] and predictions improved even more with larger families increasing evolutionary information through diversity [9, 10]. The idea was applied to other tasks, including the prediction of transmembrane regions [11–13], solvent accessibility [14], residue flexibility (B-values) [15, 16], inter-residue contacts [17] and protein disorder [15, 18–20]. Later, automatic methods predicting aspects of protein function improved by combining evolutionary information and machine learning, including predictions of subcellular localization (aka cellular compartment or CC in GO [21, 22]), protein interaction sites [23–25], and the effects of sequence variation upon function [26, 27]. Arguably, the most important breakthrough for protein structure prediction over the last decade was a more efficient way of using evolutionary couplings [28–31].

Although evolutionary information has increasingly improved prediction methods, it is also becoming increasingly costly. As sequencing becomes cheaper, the number of bio-sequence databases grow faster than computing power. For instance, the number of UniProt entries is now more than doubling every two years [32]. An all-against-all comparison executed to build up profiles of evolutionary information squares this number: every two years the job increases 4-fold while computer power grows less than 2-fold. Consequently, methods as fast as PSI-BLAST [33] have to be replaced by faster solutions such as HHblits [34]. Even its latest version HHblits3 [35] still needs several minutes to search UniRef50 (subset of UniProt) for a single query protein. The next step up in speed such as MMSeqs2 [36] appear to cope with the challenge at the expense of increasing hardware requirements while databases keep growing. However, even these solutions might eventually lose the battle against the speedup of sequencing. Analyzing data sets involving millions of proteins, i.e. samples of the human gut microbiota or metagenomic samples, have already become a major challenge [35]. Secondly, evolutionary information is still missing for some proteins, e.g. for proteins with substantial intrinsically disordered regions [15, 37, 38], or the entire *Dark Proteome* [39] full of proteins that are less-well studied but important for function [40].

Here, we propose a novel embedding of protein sequences that replaces the explicit search for evolutionary related proteins by an implicit transfer of biophysical information derived from large, unlabeled sequence data (here UniRef50). We adopted a method that has been revolutionizing Natural Language Processing (NLP), namely the bi-directional language model ELMo (Embeddings from Language Models) [41]. In NLP, ELMo is trained on unlabeled text-corpora such as Wikipedia to predict the most probable next word in a sentence, given all previous words in this sentence. By learning a probability distribution for sentences, these models autonomously develop a notion for syntax and semantics of language. The trained vector representations (embeddings) are contextualized, i.e. the embeddings of a given word depend on its context. This has the advantage that two identical words can have different embeddings, depending on the words surrounding them. In contrast to previous non-contextualized approaches such as word2-vec [42, 43], this allows to take the ambiguous meaning of words into account.

We hypothesized that the ELMo concept could be applied to model protein sequences. Three main challenges arose. (1) Proteins range from about 30 to 33,000 residues, a much larger range than for the average English sentence extending over 15–30 words [44], and even more extreme than notable literary exceptions such as James Joyce's Ulysses (1922) with almost 4000 words in a sentence. Longer proteins require more GPU memory and the underlying models (so-called LSTMs: Long Short-Term Memory networks [45]) have only a limited capability to remember long-range dependencies. (2) Proteins mostly use 20 standard amino acids, 100,000 times less tokens than in the English language. Smaller vocabularies might be problematic if protein sequences encode a similar complexity as sentences. (3) We found UniRef50 to contain almost ten times more tokens (9.5 billion amino acids) than the largest existing NLP corpus (1 billion words). Simply put: Wikipedia is roughly ten times larger than Webster's Third New International Dictionary and the entire UniProt is over ten times larger than Wikipedia. As a result, larger models might be required to absorb the information in biological databases.

We trained ELMo on UniRef50 and assessed the predictive power of the embeddings by application to tasks on two levels: per-residue (word-level) and per-protein (sentence-level). For the per-residue prediction task, we predicted secondary structure and long intrinsic disorder. For the per-protein prediction task, we predicted subcellular localization and trained a classifier distinguishing between membrane-bound and water-soluble proteins. We used publicly available data sets from two recent methods that achieved break-through performance through Deep Learning, namely NetSurfP-2.0 for secondary structure [46] and DeepLoc for localization [47]. We compared the performance of the *SeqVec* embeddings to state-of-the-art methods using evolutionary information, and also to a popular embedding tool for protein sequences originating from the Word2vec approach, namely *ProtVec* [42]. Notably, while *ProtVec* captures local information, it loses information on sequence ordering, and the resulting residue embeddings are insensitive to their context (non-contextualized), i.e. the same word results in the same embedding regardless of the specific context.

Understanding a language typically implies to understand most typical constructs convened in that language. Modeling a language in a computer can have many meanings, spanning from the automatic understanding of the semantic of languages, to parsing some underlying rules of a language (e.g. syntax). Arguably, proteins are the most important machinery of life. Protein sequence largely determines protein structure, which somehow determines protein function [48]. Thus, the expression of the language of life are essentially protein sequences. Understanding those sequences implies to predict protein structure from sequence. Despite recent successes [49, 50], this is still not possible for all proteins. However, the novel approach introduced here succeeds to model protein sequences in the sense that it implicitly extracts grammar-like principles (as embeddings) which are much more successful in predicting aspects of protein structure and function than any of the biophysical features previously used to condensate expert knowledge of protein folding, or any other previously tried simple encoding of protein sequences.

Results

Modeling protein sequences through SeqVec embeddings

SeqVec, our ELMo-based implementation, was trained for three weeks on 5 Nvidia Titan GPUs with 12 GB memory each. The model was trained until its *perplexity* (uncertainty when predicting the next token) converged at around 10.5 (Additional file 1: Figure S1). Training and testing were not split due to technical limitations (incl. CPU/GPU). ELMo was designed to reduce the risk of overfitting by sharing weights between forward and backward LSTMs and by using dropout. The model had about 93 M (mega/million) free parameters compared to the 9.6G (giga/billion) tokens to predict leading to a ratio of samples/free parameter below 1/100, the best our group has ever experienced in a prediction task. Similar approaches have shown that even todays largest models (750 M free parameters) are not able to overfit on a large corpus (250 M protein sequences) [51].

SeqVec embeddings appeared robust

When training ELMo on SWISS-PROT (0.5 M sequences), we obtained less useful models, i.e. the subsequent prediction methods based on those embeddings were less accurate. Training on UniRef50 (33 M sequences) gave significantly better results in subsequent supervised prediction tasks, and we observed similar results when using different hyperparameters. For instance, increasing the number of LSTM layers in ELMo (from two to four) gave a small, non-significant improvement. As the expansion of 2 to 4 layers roughly doubled time for training and retrieving embeddings, we decided to trade speed for insignificant improvement and continued with the faster two-layer ELMo architecture. Computational limitations hindered us from fully completeing the modelling of UniRef90 (100 million sequences). Nevertheless, after four weeks of training, the models neither appeared to be better nor significantly worse than those for UniRef50. Users of the embeddings need to be aware that every time a new ELMo model is trained, the downstream supervised prediction method needs to be retrained in the following sense. Assume we transfer-learn UniRef50 through SeqVec1, then use SeqVec1 to machine learn DeepSeqVec1 for a supervised task (e.g. localization prediction). In a later iteration, we redo the transfer learning with different hyperparameters to obtain SeqVec2. For any given sequence, the embeddings of SeqVec2 will differ from those of SeqVec1, as a result, passing embeddings derived from SeqVec2 to DeepSeqVec1 will not provide meaningful predictions.

Per-residue performance high, not highest

NetSurfP-2.0 feeds HHblits or MMseqs2 profiles into advanced combinations of Deep Learning architectures [46] to predict secondary structure, reaching a three-state per-residue accuracy Q3 of 82–85% (lower value: small, partially non-redundant CASP12 set, upper value: larger, more redundant TS115 and CB513 sets; Table 1, Fig. 1; several contenders such as *Spider3* and *RaptorX* reach within three standard errors). All six methods developed by us fell short of reaching this mark, both methods not using evolutionary information/profiles (DeepSeqVec, DeepProtVec, DeepOneHot, DeepBLOSUM65), but also those that did use profiles (*DeepProf*, DeepProf+SeqVec, Fig. 1a, Table 1). The logic in our acronyms was as follows (Methods): "*Prof*" implied using profiles (evolutionary information), *SeqVec* (Sequence-to-Vector) described using pre-trained ELMo embeddings, "Deep" before the method name suggested applying a simple deep learning method trained on particular prediction tasks using SeqVec embeddings only (DeepSeqVec), profiles without (DeepProf) or with embeddings (DeepProf+SeqVec), or other simple encoding schema (ProtVec, OneHot or sparse encoding, or BLOSUM65). When comparing methods that use only single protein sequences as input (DeepSeqVec, DeepProtVec, DeepOneHot, DeepBLOSUM65; all white in Table 1), the new method introduced here, *SeqVec* outperformed others not using profiles by three standard errors (*P*-value< 0.01; Q3: 5–10 percentage points, Q8: 5–13 percentage points, MCC:

Table 1 Per-residue predictions: secondary structure and disorder

Data	*Prediction task*	*Secondary structure*		*Disorder*	
	Method	*Q3 (%)*	*Q8 (%)*	*MCC*	*FPR*
CASP12	*NetSurfP-2.0 (hhblits)*[a,b]	**82.4**	**71.1**	0.604	**0.011**
	NetSurfP-1.0[a,b]	70.9	–	–	–
	Spider3[a,b]	79.1	–	0.582	0.026
	RaptorX[a,b]	78.6	66.1	**0.621**	0.045
	Jpred4[a,b]	76.0	–	–	–
	DeepSeqVec	73.1 ± 1.3	61.2 ± 1.6	0.575 ± 0.075	0.026 ± 0.008
	DeepProf[b]	76.4 ± 2.0	62.7 ± 2.2	0.506 ± 0.057	0.022 ± 0.009
	DeepProf + SeqVec[b]	76.5 ± 1.5	64.1 ± 1.5	0.556 ± 0.080	0.022 ± 0.008
	DeepProtVec	62.8 ± 1.7	50.5 ± 2.4	0.505 ± 0.064	0.016 ± 0.006
	DeepOneHot	67.1 ± 1.6	54.2 ± 2.1	0.461 ± 0.064	0.012 ± 0.005
	DeepBLOSUM65	67.0 ± 1.6	54.5 ± 2.0	0.465 ± 0.065	0.012 ± 0.005
TS115	*NetSurfP-2.0 (hhblits)*[a,b]	**85.3**	**74.4**	**0.663**	**0.006**
	NetSurfP-1.0[a,b]	77.9	–	–	–
	Spider3[a,b]	83.9	–	0.575	0.008
	RaptorX[a,b]	82.2	71.6	0.567	0.027
	Jpred4[a,b]	76.7	–	–	–
	DeepSeqVec	79.1 ± 0.8	67.6 ± 1.0	0.591 ± 0.028	0.012 ± 0.001
	DeepProf[b]	81.1 ± 0.6	68.3 ± 0.9	0.516 ± 0.028	0.012 ± 0.002
	DeepProf + SeqVec[b]	82.4 ± 0.7	70.3 ± 1.0	0.585 ± 0.029	0.013 ± 0.003
	DeepProtVec	66.0 ± 1.0	54.4 ± 1.3	0.470 ± 0.028	0.011 ± 0.002
	DeepOneHot	70.1 ± 0.8	58.5 ± 1.1	0.476 ± 0.028	0.008 ± 0.001
	Deep BLOSUM65	70.3 ± 0.8	58.1 ± 1.1	0.488 ± 0.029	0.007 ± 0.001
CB513	*NetSurfP-2.0 (hhblits)*[a,b]	**85.3**	**72.0**	–	–
	NetSurfP-1.0[a,b]	78.8	–	–	–
	Spider3[a,b]	84.5	–	–	–
	RaptorX[a,b]	82.7	70.6	–	–
	Jpred4[a,b]	77.9	–	–	–
	DeepSeqVec	76.9 ± 0.5	62.5 ± 0.6	–	–
	DeepProf[b]	80.2 ± 0.4	64.9 ± 0.5	–	–
	DeepProf + SeqVec[b]	80.7 ± 0.5	66.0 ± 0.5	–	–
	DeepProtVec	63.5 ± 0.4	48.9 ± 0.5	–	–
	DeepOneHot	67.5 ± 0.4	52.9 ± 0.5	–	–
	DeepBLOSUM65	67.4 ± 0.4	53.0 ± 0.5	–	–

Performance comparison for secondary structure (3- vs. 8-classes) and disorder prediction (binary) for the CASP12, TS115 and CB513 data sets. Accuracy (Q3, Q10) is given in percentage. Results marked by [a] are taken from NetSurfP-2.0 [46]; the authors did not provide standard errors. Highest numerical values in each column in bold letters. Methods DeepSeqVec, DeepProtVec, DeepOneHot and DeepBLOSUM65 use only information from single protein sequences. Methods using evolutionary information (MSA profiles) are marked by [b]; these performed best throughout

0.07–0.12, Table 1). Using a context-independent language model derived from the Word2vec approach, namely DeepProtVec was worse by 10 percentage points (almost six standard errors). On the other hand, our implementation of evolutionary information (DeepProf using HHblits profiles) remained about 4–6 percentage points below NetSurfP-2.0 (Q3 = 76–81%, Fig. 1, Table 1). Depending on the test set, using *SeqVec* embeddings instead of evolutionary information (DeepSeqVec: Fig. 1a, Table 1) remained 2–3 percentage points below that mark (Q3 = 73–79%, Fig. 1a, Table 1). Using both evolutionary information and *SeqVec* embeddings (DeepProf+SeqVec) improved over both, but still did not reach the top (Q3 = 77–82%). In fact, the ELMo embeddings alone (DeepSeqVec) did not surpass any of the best methods using evolutionary information tested on the same data set (Fig. 1a).

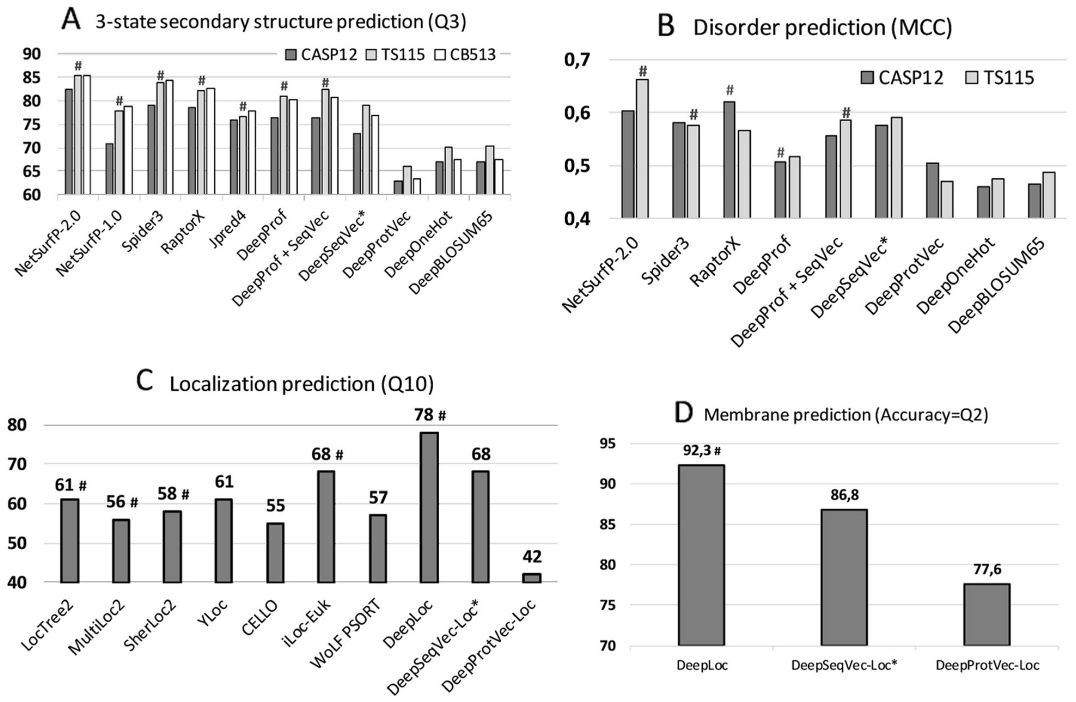

Fig. 1 Performance comparisons. The predictive power of the ELMo-based SeqVec embeddings was assessed for per-residue (upper row) and per-protein (lower row) prediction tasks. Methods using evolutionary information are highlighted by hashes above the bars. Approaches using only the proposed *SeqVec* embeddings are highlighted by stars after the method name. Panel **A** used three different data sets (CASP12, TS115, CB513) to compare three-state secondary structure prediction (y-axis: Q3; all DeepX developed here to test simple deep networks on top of the encodings tested; DeepProf used evolutionary information). Panel **B** compared predictions of intrinsically disordered residues on two data sets (CASP12, TS115; y-axis: MCC). Panel **C** compared per-protein predictions for subcellular localization between top methods (numbers for Q10 taken from DeepLoc [47]) and embeddings based on single sequences (Word2vec-like *ProtVec* [42] and our ELMo-based *SeqVec*). Panel **D**: the same data set was used to assess the predictive power of SeqVec for the classification of a protein into membrane-bound and water-soluble

For the prediction of intrinsic disorder, we observed the same: NetSurfP-2.0 performed best; our implementation of evolutionary information (DeepProf) performed worse (Fig. 1b, Table 1). However, for this task the embeddings alone (DeepSeqVec) performed relatively well, exceeding our in-house implementation of a model using evolutionary information (DeepSeqVec MCC = 0.575–0.591 vs. DeepProf MCC = 0.506–0.516, Table 1). The combination of evolutionary information and embeddings (DeepProf+SeqVec) improved over using evolutionary information alone but did not improve over the *SeqVec* embeddings for disorder. Compared to other methods, the embeddings alone reached similar values (Fig. 1b).

Per-protein performance close to best

For predicting subcellular localization (cellular compartments) in ten classes, *DeepLoc* [47] is top with Q10 = 78% (Fig. 1c, Table 2). For simplicity, we only tested methods not using evolutionary information/profiles for this task. Our sequence-only embeddings model DeepSeqVec-Loc reached second best performance together with iLoc-Euk [52] at Q10 = 68% (Fig. 1c, Table 2). Unlike the per-residue predictions, for this application the SeqVec embeddings outperformed several popular prediction methods that use evolutionary information by up to 13 percentage points in Q10 (Table 2: DeepSeqVec-Loc vs. methods shown in grayed rows). The gain of the context-dependent SeqVec model introduced here over context-independent versions such as ProtVec (from Word2vec) was even more pronounced than for the per-residue prediction task (Q10 68 ± 1% vs. 42 ± 1%).

Performance for the classification into membrane-bound and water-soluble proteins followed a similar trend (Fig. 1d, Table 2): while DeepLoc still performed best (Q2 = 92.3, MCC = 0.844), DeepSeqVec-Loc reached just a few percentage points lower (Q2 = 86.8 ± 1.0,

Table 2 Per-protein predictions: localization and membrane/globular

Method	*Localization*		*Membrane/globular*	
	Q10 (%)	*Gorodkin (MCC)*	*Q2*	*MCC*
LocTree2[a,b]	61	0.53		
MultiLoc2[a,b]	56	0.49		
CELLO[a]	55	0.45		
WoLF PSORT[a]	57	0.48		
YLoc[a]	61	0.53		
SherLoc2[a,b]	58	0.51		
iLoc-Euk[a,b]	68	0.64		
DeepLoc[a,b]	**78**	**0.73**	**92.3**	**0.844**
DeepSeqVec-Loc	68 ± 1	0.61 ± 0.01	86.8 ± 1.0	0.725 ± 0.021
DeepProtVec-Loc	42 ± 1	0.19 ± 0.01	77.6 ± 1.3	0.531 ± 0.026

Performance for per-protein prediction of subcellular localization and classifying proteins into membrane-bound and water-soluble. Results marked by [a] taken from DeepLoc [47]; the authors provided no standard errors. The results reported for *SeqVec* and *ProtVec* were based on single protein sequences, i.e. methods NOT using evolutionary information (neither during training nor testing). All methods using evolutionary information are marked by [b]; best in each set marked by bold numbers

MCC = 0.725 ± 0.021; full confusion matrix Additional file 1: Figure S2). In contrast to this, ProtVec, another method using only single sequences, performed substantially worse (Q2 = 77.6 ± 1.3, MCC = 0.531 ± 0.026).

Visualizing results

Lack of insight often triggers the misunderstanding that machine learning methods are black box solutions barring understanding. In order to interpret the *SeqVec* embeddings, we have projected the protein-embeddings of the per-protein prediction data upon two dimensions using t-SNE [53]. We performed this analysis once for the raw embeddings (SeqVec, Fig. 2 upper row) and once for the hidden layer representation of the per-protein network (DeepSeqVec-Loc) after training (Fig. 2 lower row). All t-SNE representations in Fig. 2 were created using 3000 iterations and the cosine distance as metric. The two analyses differed only in that the perplexity was set to 20 for one (*SeqVec*) and 15 for the other (DeepSeqVec-Loc). The t-SNE representations were colored either according to their localization within the cell (left column of Fig. 2) or according to whether they are membrane-bound or water-soluble (right column).

Despite never provided during training, the raw embeddings appeared to capture some signal for classifying proteins by localization (Fig. 2, upper row, left column). The most consistent signal was visible for extra-cellular proteins. Proteins attached to the cell membrane or located in the endoplasmic reticulum also formed well-defined clusters. In contrast, the raw embeddings neither captured a consistent signal for nuclear nor for mitochondrial proteins. Through training, the network improved the signal to reliably classify mitochondrial and plastid proteins. However, proteins in the nucleus and cell membrane continued to be poorly distinguished via t-SNE.

Coloring the t-SNE representations for membrane-bound or water-soluble proteins (Fig. 2, right column), revealed that the raw embeddings already provided well-defined clusters although never trained on membrane prediction (Fig. 2, upper row). After training, the classification was even better (Fig. 2, lower row).

Analogously, we used t-SNE projections to analyze SeqVec embeddings on different levels of complexity inherent to proteins (Fig. 3), ranging from the building blocks (amino acids, Fig. 3a), to secondary structure defined protein classes (Fig. 3b), over functional features (Fig. 3c), and onto the macroscopic level of the kingdoms of life and viruses (Fig. 3d; classifications in panels 3b-3d based on SCOPe [54]). Similar to the results described in [51], our projection of the embedding space confirmed that the model successfully captured bio-chemical and bio-physical properties on the most fine-grained level, i.e. the 20 standard amino acids (Fig. 3a). For example, aromatic amino acids (W, F, Y) are well separated from aliphatic amino acids (A, I, L, M, V) and small amino acids (A, C, G, P, S, T) are well separated from large ones (F, H, R, W, Y). The projection of the letter indicating an unknown amino acid (X), clustered closest to the amino acids alanine (A) and glycine (G) (data not shown). Possible explanations for this could be that the two amino acids with the smallest side chains might be least biased towards other biochemical features like charge and that they are the 2nd (A) and 4th (G) most frequent amino acids in our training set (Additional file 1: Table S1). Rare (O, U) and ambiguous amino acids (Z, B) were removed from the projection as their clustering showed that the model could not learn reasonable embeddings from the very small number of samples.

High-level structural classes as defined in SCOPe (Fig. 3b) were also captured by SeqVec embeddings. Although the embeddings were only trained to predict the next amino acid in a protein sequence, well separated clusters emerged from those embeddings in structure space. Especially, membrane proteins and small proteins formed distinct clusters (note: protein length is not explicitly encoded in *SeqVec*). Also, these results indicated that the embeddings captured complex relationships between proteins which are not directly observable from sequence similarity alone as SCOPe was redundancy reduced at 40% sequence identity. Therefore, the new embeddings could complement sequence-based structural classification as it was shown that the sequence similarity does not necessarily lead to structural similarity [55].

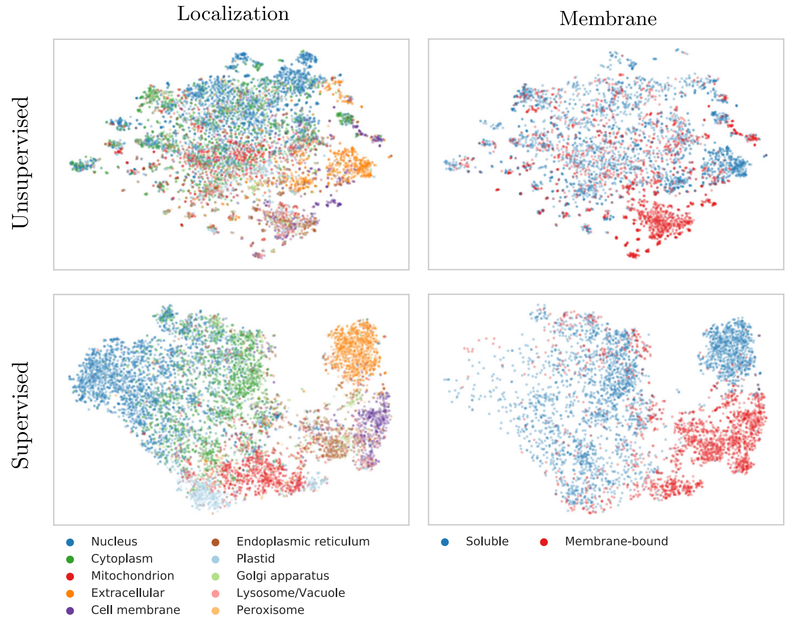

Fig. 2 t-SNE representations of SeqVec. Shown are t-SNE projections from embedded space onto a 2D representation; upper row: unsupervised 1024-dimensional "raw" ELMo-based SeqVec embeddings, averaged over all residues in a protein; lower row: supervised 32-dimensional ELMo-based SeqVec embeddings, reduced via per-protein machine learning predictions (data: redundancy reduced set from DeepLoc). Proteins were colored according to their localization (left column) or whether they are membrane-bound or water-soluble (right column). Left and right panel would be identical except for the color, however, on the right we had to leave out some points due to lacking membrane/non-membrane annotations. The upper row suggests that *SeqVec* embeddings capture aspects of proteins without ever seeing labels of localization or membrane, i.e. without supervised training. After supervised training (lower row), this information is transferred to, and further distilled by networks with simple architectures. After training, the power of SeqVeq embeddings to distinguish aspects of function and structure become even more pronounced, sometimes drastically so, as suggested by the almost fully separable clusters in the lower right panel

To further investigate the clusters emerging from the SCOPe data set, we colored the same data set based on protein functions (Fig. 3c) and kingdoms (Fig. 3d). This analysis revealed that many of the small, distinct clusters emerged based on protein functions. For instance, transferases and hydrolases formed many small clusters. When increasing the level of abstraction by coloring the proteins according to their kingdoms, we observed certain clusters to be dominated by e.g. eukaryotes. Comparing the different views captured in panels 3B-3D revealed connections, e.g. that all-beta or small proteins dominate in eukaryotes (compare blue and orange islands in Fig. 3b with the same islands in Fig. 3d – colored blue to mark eukaryotes).

CPU/GPU time used

Due to the sequential nature of LSTMs, the time required to embed a protein grows linearly with protein length. Depending on the available main memory or GPU memory, this process could be massively parallelized. To optimally use available memory, batches are typically based on tokens rather than on sentences. In order to retrieve embeddings, we sorted proteins according to their length and created batches of ≤15 K tokens that could still be handled by a single Nvidia GeForce GTX1080 with 8GB VRAM. The processing of a single protein took on average 0.027 s when applying this batch-strategy to the NetSurfP-2.0 data set (average protein length: 256 residues, i.e. shorter than proteins for which 3D structure is not known). The batch with the shortest proteins (on average 38 residues, corresponding to 15% of the average protein length in the whole data set) required about one tenth (0.003 s per protein, i.e. 11% of that for whole set). The batch containing the longest protein sequences in this data set (1578 residues on average, corresponding to 610% of average protein length in the whole data set), took about six times more (1.5 s per protein, i.e. 556% of that for whole set). When creating SeqVec for the DeepLoc set (average length: 558 residues; as this set does not require

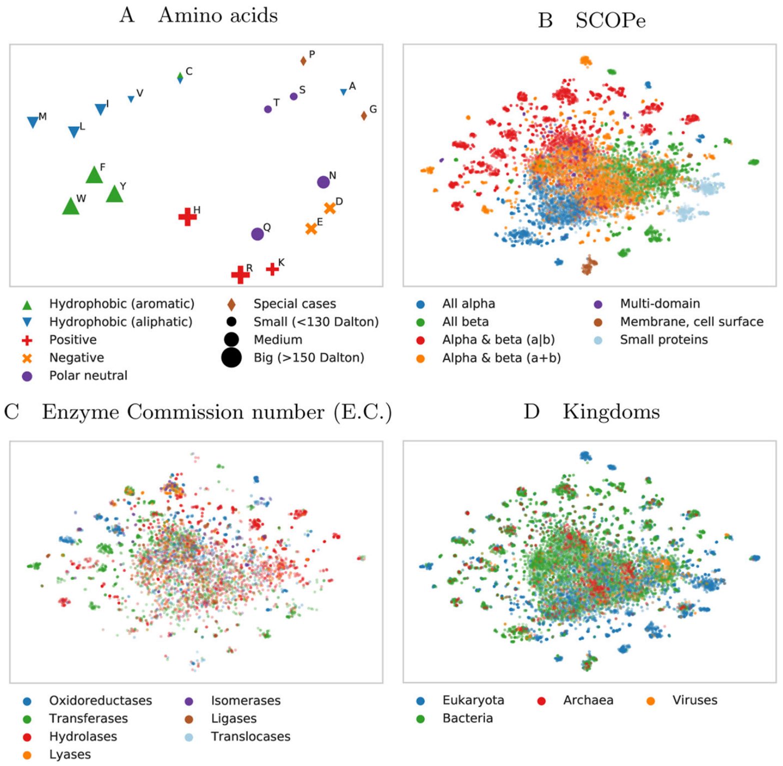

Fig. 3 Modeling aspects of the language of life. 2D t-SNE projections of unsupervised *SeqVec* embeddings highlight different realities of proteins and their constituent parts, amino acids. Panels **B** to **D** are based on the same data set (Structural Classification of Proteins – extended (SCOPe) 2.07, redundancy reduced at 40%). For these plots, only subsets of SCOPe containing proteins with the annotation of interest (enzymatic activity C and kingdom D) may be displayed. Panel **A**: the embedding space confirms: the 20 standard amino acids are clustered according to their biochemical and biophysical properties, i.e. hydrophobicity, charge or size. The unique role of Cysteine (C, mostly hydrophobic and polar) is conserved. Panel B: SeqVec embeddings capture structural information as annotated in the main classes in SCOPe without ever having been explicitly trained on structural features. Panel **C**: many small, local clusters share function as given by the main classes in the Enzyme Commission Number (E.C.). Panel **D**: similarly, small, local clusters represent different kingdoms of life

a 3D structure, it provides a more realistic view on the distribution of protein lengths), the average processing time for a single protein was 0.08 with a minimum of 0.006 for the batch containing the shortest sequences (67 residues on average) and a maximum of 14.5 s (9860 residues on average). On a single Intel i7–6700 CPU with 64GB RAM, processing time increased by roughly 50% to 0.41 s per protein, with a minimum and a maximum computation time of 0.06 and 15.3 s, respectively. Compared to an average processing time of one hour for 1000 proteins when using evolutionary information directly [46], this implied an average speed up of 120-fold on a single GeForce GTX1080 and 9-fold on a single i7–6700 when predicting structural features; the inference time of DeepSeqVec for a single protein is on average 0.0028 s.

Discussion

Transfer-learning alone not top

The context-dependent transfer-learning model ELMo [41] applied to proteins sequences (here dubbed *SeqVec*) clearly succeeded to model the language of protein sequences much better than simple schema (e.g. one-hot encoding), more advanced context-independent language models such as ProtVec (based on Word2vec [42, 43]), more advanced distillations of text-book knowledge (biophysical features used as input for prediction [2, 3]), and also some family-independent information about evolution as represented by the expertise condensed in the BLOSSUM62 matrix. In this sense, our approach worked. However, none of our SeqVec implementations reached today's best methods: NetSurfP-2.0 for secondary structure and protein disorder and DeepLoc for localization and membrane protein classification (Fig. 1, Table 1, Table 2). Clearly, "just" using SeqVec embeddings to train subsequent prediction methods did not suffice to crack the challenges. Due to computational limitations, testing models trained on larger sequence database, which may over-come this limitation, could not be tested. What about more advanced transfer-learning models, e.g. TransformerXL [56], or different pre-training objectives which model bidirectional contexts, e.g. Bert [57] or XLNet [58]? We have some evidence that transformer-based models might reach further (Elnaggar et al. in preparation), with competing groups already showing promising results [51]. Nevertheless, there is one major reality to remember: we model single protein sequences. Such models might learn the rules for "writing protein sequences" and still miss the constraints imposed by the "survival of the fittest", i.e. by evolutionary selection.

On the other hand, some of our solutions appeared surprisingly competitive given the simplicity of the architectures. In particular, for the per-protein predictions, for which *SeqVec* clearly outperformed the previously popular *ProtVec* [42] approach and even commonly used expert solutions (Fig. 1, Table 2: no method tested other than the top-of-the-line *DeepLoc* reached higher numerical values). For that comparison, we used the same data sets but could not rigorously compare standard errors (SE) that were unavailable for other methods. Estimating standard errors for our methods suggested the differences to be statistically significant: > 7 SE throughout (exception: DeepLoc (Q10 = 78) and iLoc-Euk(Q10 = 68)). The results for localization prediction implied that frequently used methods using evolutionary information (all marked with shaded boxes in Table 2) did not clearly outperform our simple ELMo-based tool (DeepSeqVec-Loc in Table 2). This was very different for the per-residue prediction tasks: here almost all top methods using evolutionary information numerically outperformed the simple model built on the ELMo embeddings (DeepSeqVec in Fig. 1 and Table 1). However, all models introduced in this work were deliberately designed to be relatively simple to demonstrate the predictive power of *SeqVec*. More sophisticated architectures building up on *SeqVec* embeddings will likely outperform the approaches introduced here.

Combining SeqVec with evolutionary information for per-residue predictions still did not reach the top (set TS115: Q3(NetSurfP-2.0) = 85.3% vs. Q3(DeepProf + SeqVec) = 82.4%, Table 1). This might suggest some limit for the usefulness of the ELMo-based SeqVec embeddings. However, it might also point to the more advanced solutions realized by NetSurfP-2.0 which applies two LSTMs of similar complexity as our entire system (including ELMo) on top of their last step leading to 35 M (35 million) free parameters compared to about 244 K for DeepProf + SeqVec. Twenty times more free parameters might explain some fraction of the success. Due to limited GPU resources, we could not test how much.

Why did the ELMo-based approach improve more (relative to competition) for per-protein than for per-residue predictions? We can only speculate because none of the possible explanations have held consistently for all methods to which we have been applying ELMo embeddings over the recent six months (data not shown). For instance, the per-protein data sets were over two orders of magnitude smaller than those for per-residue predictions; simply because every protein constitutes one sample in the first and protein length samples for the second. SeqVec might have helped more for the smaller data sets because the unlabeled data is pre-processed so meaningful that less information needs to be learned by the ANN during per-protein prediction. This view was strongly supported by the t-SNE [53] results (Fig. 2, Fig. 3): ELMo apparently had learned the "grammar" of the language of life well enough to realize a very rough clustering of structural classes, protein function, localization and membrane/not. Another, yet complementary, explanation for this trend could be that the training of ELMo inherently provides a natural way of summarizing information of proteins of varying length. Other approaches usually learn this summarization step together with the actual prediction tasks which gets increasingly difficult the smaller the data set.

We picked four tasks as proof-of-principle for our ELMo/SeqVec approach. These tasks were picked because recent breakthroughs had been reported (e.g. NetSurfP-2.0 [46] and DeepLoc [47]) and those had made data for training and testing publicly available. We cannot imagine why our findings should not hold true for other tasks of protein prediction and invite the community to apply the *SeqVec* embeddings for their tasks. We assume the SeqVec embeddings to be more beneficial for small than for large data sets. For instance, we expect little or no gain in predicting inter-residue contacts, and more in predicting protein binding sites.

Good and fast predictions without using evolutionary information

Although our SeqVec embeddings were over five percentage points worse than the best method NetSurfP-2.0 (Table 1: TS115 Q3: 85.3 vs. 79.1), for some proteins (12% in CB513) DeepSeqVec performed better (Additional file 1: Figure S4). We expect those to be proteins with small or incorrect alignments, however, due to the fact that we did not have the alignments available used by NetSurfP-2.0, we could not quite establish the validity of this assumption (analyzing pre-computed alignments from ProteinNet [59] revealed no clear relation of the type: more evolutionary information leads to better prediction). However, the real strength of our solutions is its speed: SeqVec predicted secondary structure and protein disorder over 100-times faster (on a single 8GB GPU) than NetSurfP-2.0 when counting the time it needs to retrieve the evolutionary information summarized in alignment profiles although using the fastest available alignment method, namely MMseqs2 [36] which already can reach speed-up values of 100-times over PSI-BLAST [33]. For those who do not have enough resources for running MMSeqs2 and therefore have to rely on PSI-BLAST, the speed-up of our prediction becomes 10,000-fold. Even the 100-fold speed-up is so substantial that for some applications, the speedup might outweigh the reduction in performance. Embedding based approaches such as *SeqVec* suggest a promising solution toward solving one of the biggest challenges for computational biology: how to efficiently handle the exponentially increasing number of sequences in protein databases? Here, we showed that relevant information from large unannotated biological databases can be compressed into embeddings that condense and abstract the underlying biophysical principles. These embeddings, essentially the weights of a neural network, help as input to many problems for which smaller sets of annotated data are available (secondary structure, disorder, localization). Although the compression step needed to build the *SeqVec* model is very GPU-intensive, it can be performed in a centralized way using large clusters. After training, the model can be shipped and used on any consumer hardware. Such solutions are ideal to support researches without access to expensive cluster infrastructure.

Modeling the language of life?

SeqVec, our pre-trained ELMo adaption, learned to model a probability distribution over a protein sequence. The sum over this probability distribution constituted a very informative input vector for any machine learning task trying to predict protein features. It also picked up context-dependent protein motifs without explicitly explaining what these motifs are relevant for. In contrast, context-independent tools such as *ProtVec* [42] will always create the same vectors regardless of the residues surrounding this k-mer in a protein sequence.

Our hypothesis had been that the ELMo-based *SeqVec* embeddings trained on large databases of un-annotated protein sequences could extract a *probabilistic model of the language of life* in the sense that the resulting system will extract aspects relevant both for per-residue and per-protein prediction tasks. All results presented here have added independent evidence in full support of this hypothesis. For instance, the three state per-residue accuracy for secondary structure prediction improved by over eight percentage points through ELMo (Table 1, e.g. Q3: 79.1 vs. 70.3%), the per-residue MCC for protein disorder prediction also increased substantially (Table 1, e.g. MCC: 0.591 vs. 0.488). On the per-protein level, the improvement over the previously popular tool extracting “meaning” from proteins, *ProtVec*, was even more substantial (Table 1: e.g. Q10: 68% vs. 42%). We could demonstrate this reality even more directly using the t-SNE [53] results (Fig. 2 and Fig. 3): different levels of complexity ranging from single amino acids, over some localizations, structural features, functions and the classification of membrane/non-membrane had been implicitly learned by *SeqVec* without training. Clearly, our ELMo-driven implementation of transfer-learning fully succeeded to model some aspects of the language of life as proxied by protein sequences. How much more will be possible? Time will tell.

Conclusion

We have shown that it is possible to capture and transfer knowledge, e.g. biochemical or biophysical properties, from a large unlabeled data set of protein sequences to smaller, labelled data sets. In this first proof-of-principle, our comparably simple models have already reached promising performance for a variety of per-residue and per-protein prediction tasks obtainable from only single protein sequences as input, that is: without any direct evolutionary information, i.e. without profiles from multiple sequence alignments of protein families. This reduces the dependence on the time-consuming and computationally intensive calculation of protein profiles, allowing the prediction of per-residue and per-protein features of a whole proteome within less than an hour. For instance, on a single GeForce GTX 1080, the creation of embeddings and predictions of secondary structure and subcellular localization for the whole human proteome took about 32 min. Building more sophisticated architectures on top of *SeqVec* might increase sequence-based performance further.

Our new *SeqVec* embeddings may constitute an ideal starting point for many different applications in particular when labelled data are limited. The embeddings combined with evolutionary information might even improve over the best available methods, i.e. enable high-quality predictions. Alternatively, they might ease high-throughput predictions of whole proteomes when used as the only input feature. Alignment-free predictions bring speed and improvements for proteins for which alignments are not readily available or limited, such as for intrinsically disordered proteins, for the Dark Proteome, or for particular unique inventions of evolution. The trick was to tap into the potential of Deep

Learning through transfer learning from large repositories of unlabeled data by modeling the language of life.

Methods

Data

UniRef50 training of *SeqVec:* We trained ELMo on UniRef50 [32], a sequence redundancy-reduced subset of the UniProt database clustered at 50% pairwise sequence identity (PIDE). It contained 25 different letters (20 standard and 2 rare amino acids (U and O) plus 3 special cases describing either ambiguous (B, Z) or unknown amino acids (X); Additional file 1: Table S1) from 33 M proteins with 9,577,889,953 residues. In order to train ELMo, each protein was treated as a sentence and each amino acid was interpreted as a single word.

Visualization of embedding space: The current release of the "Structural Classification Of Proteins" (SCOPe, [54]) database (2.07) contains 14,323 proteins at a redundancy level of 40%. Functions encoded by the Enzyme Commission number (E.C., [60]) were retrieved via the "Structure Integration with Function, Taxonomy and Sequence" (SIFTS) mapping [61]. SIFTS allows, among other things, a residue-level mapping between UniProt and PDB entries and a mapping from PDB identifiers to E.C.s. If no function annotation was available for a protein or if the same PDB identifier was assigned to multiple E.C.s, it was removed from Fig. 3c. Taxonomic identifiers from UniProt were used to map proteins to one of the 3 kingdoms of life or to viruses. Again, proteins were removed if no such information was available. The number of iterations for the t-SNE projections was set again to 3000 and the perplexity was adjusted (perplexity = 5 for Fig. 3a and perplexity = 30 for Fig. 3b-d).

Per-residue level: secondary structure & intrinsic disorder (*NetSurfP-2.0*). To simplify comparability, we used the data set published with a recent method seemingly achieving the top performance of the day in secondary structure prediction, namely *NetSurfP-2.0* [46]. Performance values for the same data set exist also for other recent methods such as *Spider3* [62], *RaptorX* [63, 64] and *JPred4* [65]. The set contains 10,837 sequence-unique (at 25% PIDE) proteins of experimentally known 3D structures from the PDB [66] with a resolution of 2.5 Å (0.25 nm) or better, collected by the PISCES server [67]. DSSP [68] assigned secondary structure and intrinsically disordered residues are flagged (residues without atomic coordinates, i.e. REMARK-465 in the PDB file). The original seven DSSP states (+ 1 for unknown) were mapped upon three states using the common convention: [G,H, I] → H (helix), [B,E] → E (strand), all others to O (other; often misleadingly referred to as *coil* or *loop*). As the authors of NetSurfP-2.0 did not include the raw protein sequences in their public data set, we used the SIFTS file to obtain the original sequence. Only proteins with identical length in SIFTS and NetSurfP-2.0 were used. This filtering step removed 56 sequences from the training set and three from the test sets (see below: two from CB513, one from CASP12 and none from TS115). We randomly selected 536 (~ 5%) proteins for early stopping (*cross-training*), leaving 10,256 proteins for training. All published values referred to the following three test sets (also referred to as validation set): **TS115** [69]: 115 proteins from high-quality structures (< 3 Å) released after 2015 (and at most 30% PIDE to any protein of known structure in the PDB at the time); **CB513** [70]: 513 non-redundant sequences compiled 20 years ago (511 after SIFTS mapping); **CASP12** [71]: 21 proteins taken from the CASP12 free-modelling targets (20 after SIFTS mapping; all 21 fulfilled a stricter criterion toward non-redundancy than the two other sets; non-redundant with respect to all 3D structures known until May 2018 and all their relatives). Each of these sets covers different aspects of the secondary structure prediction problem: CB513 and TS115 only use structures determined by X-ray crystallography and apply similar cutoffs with respect to redundancy (30%) and resolution (2.5–3.0 Å). While these serve as a good proxy for a baseline performance, CASP12 might better reflect the true generalization capability for unseen proteins as it includes structures determined via NMR and Cryo-EM. Also, the strict redundancy reduction based on publication date reduces the bias towards well studied families. Nevertheless, toward our objective of establishing a proof-of-principle, these sets sufficed. All test sets had fewer than 25% PIDE to any protein used for training and cross-training (ascertained by the *NetSurfP-2.0* authors). To compare methods using evolutionary information and those using our new word embeddings, we took the *HHblits* profiles published along with the NetSurfP-2.0 data set.

Per-protein level: subcellular localization & membrane proteins (DeepLoc). Subcellular localization prediction was trained and evaluated using the *DeepLoc* data set [47] for which performance was measured for several methods, namely: LocTree2 [72], MultiLoc2 [73], SherLoc2 [74], CELLO [75], iLoc-Euk [52], WoLF PSORT [76] and YLoc [77]. The data set contained proteins from UniProtKB/Swiss-Prot [78] (release: 2016_04) with experimental annotation (code: ECO:0000269). The *DeepLoc* authors mapped these annotations to ten classes, removing all proteins with multiple annotations. All these proteins were also classified into *water-soluble* or *membrane-bound* (or as *unknown* if the annotation was ambiguous). The resulting 13,858 proteins were clustered through PSI-CD-HIT [79, 80] (version 4.0; at 30% PIDE or Eval< 10^{-6}). Adding the requirement that the alignment had to cover 80% of the shorter protein, yielded 8464 clusters. This set was split into training and testing by using the same proteins for testing as the authors of DeepLoc. The training set was randomly subdivided into 90% for training and 10% for determining early stopping (cross-training set).

Embedding terminology and related work

One-hot encoding (also known as *sparse encoding*) assigns each word (referred to as token in NLP) in the vocabulary an integer N used as the Nth component of a vector with the dimension of the vocabulary size (number of different words). Each component is binary, i.e. either 0 if the word is not present in a sentence/text or 1 if it is. This encoding drove the first application of machine learning that clearly improved over all other methods in protein prediction [1–3]. TF-IDF represents tokens as the product of "frequency of token in data set" times "inverse frequency of token in document". Thereby, rare tokens become more relevant than common words such as "the" (so called *stop words*). This concept resembles that of using k-mers for database searches [33], clustering [81], motifs [82, 83], and prediction methods [72, 76, 84–88]. Context-insensitive word embeddings replaced expert features, such as TF-IDF, by algorithms that extracted such knowledge automatically from unlabeled corpus such as Wikipedia, by either predicting the neighboring words, given the center word (skip-gram) or vice versa (CBOW). This became known in *Word2Vec* [43] and showcased for computational biology through *ProtVec* [43, 89]. ProtVec assumes that every token or word consists of three consecutive residues (amino acid 3-mers). During training, each protein sequence in *SwissProt* [78] is split into overlapping 3-mers and the skip-gram version of *word2vec* is used to predict adjacent 3-mers, given the 3-mer at the center. After training, protein sequences can be split into overlapping 3-mers which are mapped onto a 100-dimensional latent space. More specialized implementations are *mut2vec* [90] learning mutations in cancer, and *phoscontext2vec* [91] identifying phosphorylation sites. Even though the performance of context-insensitive approaches was pushed to its limits by adding sub-word information (FastText [92]) or global statistics on word co-occurance (GloVe [93]), their expressiveness remained limited because the models inherently assigned the same vector to the same word, regardless of its context. Context-sensitive word embeddings started a new wave of word embedding techniques for NLP in 2018: the embedding renders the meaning of words and phrases such as *"paper tiger"* dependent upon the context, allowing to account for the ambiguous meanings of words. Popular examples like ELMo [41] and Bert [57] have achieved state-of-the-art results in several NLP tasks. Both require substantial GPU computing power and time to be trained from scratch. One of the main differences between ELMo and Bert is their pre-training objective: while auto-regressive models like ELMo predict the next word in a sentence given all previous words, autoencoder-based models like Bert predict masked-out words given all words which were not masked out. However, in this work we focused on ELMo as it allows processing of sequences of variable length. The original ELMo model consists of a single, context-insensitive CharCNN [94] over the characters in a word and two layers of bidirectional LSTMs that introduce the context information of surrounding words (Fig. 4). The CharCNN transforms all characters within a single word via an embedding layer into vector space and runs multiple CNNs of varying window size (here: ranging from 1 to 7) and number of filters (here: 32, 64, …, 1024). In order to obtain a fixed-dimensional vector for each word, regardless of its length, the output of the CNNs is max-pooled and concatenated. This feature is crucial for NLP in order to be able to process words of variable length. As our words consist only of single amino acids, this layer learns an uncontextualized mapping of single amino acids onto a latent space. The first bi-directional LSTM operates directly on the output of the CharCNN, while the second LSTM layer takes the output of the first LSTM as input. Due to their sequential nature, the LSTM layers render the embeddings dependent on their context as their internal state always depends on the previous hidden state. However, the bidirectionality of the LSTMs would lead to information leakage, rendering the training objective trivial, i.e. the backward pass had already seen the word which needs to be predicted in the forward pass. This problem is solved by training the forward and the backward pass of the LSTMs independently, i.e. the forward pass is conditioned only on words to its left and vice versa. During inference the internal states of both directions are concatenated allowing the final embeddings to carry information from both sides of the context. As described in the original ELMo publication, the weights of the forward and the backward model are shared in order to reduce the memory overhead of the model and to combat overfitting. Even though, the risk of overfitting is small due to the high imbalance between number of trainable parameters (93 M) versus number of tokens (9.3B), dropout at a rate of 10% was used to reduce the risk of overfitting. This model is trained to predict the next amino acid given all previous amino acids in a protein sequence. To the best of our knowledge, the context-sensitive ELMo has not been adapted to protein sequences, yet.

ELMo adaptation

In order to adapt ELMo [41] to protein sequences, we used the standard ELMo configuration with the following changes: (i) reduction to 28 tokens (20 standard and

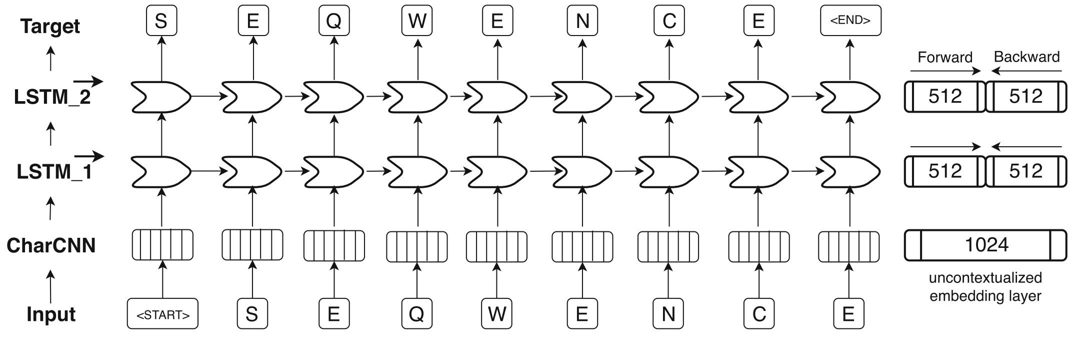

Fig. 4 ELMo-based architecture adopted for SeqVec. First, an input sequence, e.g. "S E Q W E N C E" (shown at bottom row), is padded with special tokens indicating the start ("<start>") and the end ("<end>") of the sentence (here: protein sequences). On the 2nd level (2nd row from bottom), character convolutions (CharCNN, [94]) map each word (here: amino acid) onto a fixed-length latent space (here: 1024-dimensional) without considering information from neighboring words. On the third level (3rd row from bottom), the output of the CharCNN-layer is used as input by a bidirectional Long Short Term Memory (LSTM, [45]) which introduces context-specific information by processing the sentence (protein sequence) sequentially. For simplicity, only the forward pass of the bi-directional LSTM-layer is shown (here: 512-dimensional). On the fourth level (4th row from bottom), the second LSTM-layer operates directly on the output of the first LSTM-layer and tries to predict the next word given all previous words in a sentence. The forward and backward pass are optimized independently during training in order to avoid information leakage between the two directions. During inference, the hidden states of the forward and backward pass of each LSTM-layer are concatenated to a 1024-dimensional embedding vector summarizing information from the left and the right context

2 rare (U,O) amino acids + 3 special tokens describing ambiguous (B,Z) or unknown (X) amino acids + 3 special tokens for ELMo indicating padded elements ('< MASK>') or the beginning ('<S>') or the end of a sequence ('</S>')), (ii) increase number of unroll steps to 100 to account for the increased length of protein sequences compared to sentences in natural languages, (iii) decrease number of negative samples to 20, (iv) increase token number to 9,577,889,953. After pre-training the ELMo architecture (1 CharCNN, 2 LSTM-Layers, see "Embedding terminology and related work" section and Fig. 4 for more details) with our parameters on UniRef50, the embedding model takes a protein sequence of arbitrary length and returns 3076 features for each residue in the sequence. These 3076 features were derived by concatenating the outputs of the three layers of ELMo, each describing a token with a vector of length 1024. The LSTM layers were composed of the embedding of the forward pass (first 512 dimensions) and the backward pass (last 512 dimensions). In order to demonstrate the general applicability of ELMo or *SeqVec* and to allow for easy integration into existing models, we neither fine-tuned the pre-trained model on a specific prediction task, nor optimized the combination of the three internal layers. Thus, researchers could just replace (or concatenate) their current machine learning inputs with our embeddings to boost their task-specific performance. Furthermore, it will simplify the development of custom models that fit other use-cases. For simplicity, we summed the components of the three 1024-dimensional vectors to form a single 1024-dimensional feature vector describing each residue in a protein.

Using SeqVec for predicting protein features

On the per-residue level, the predictive power of the new *SeqVec* embeddings was demonstrated by training a small two-layer Convolutional Neural Network (CNN) in PyTorch using a specific implementation [95] of the ADAM optimizer [96], cross-entropy loss, a learning rate of 0.001 and a batch size of 128 proteins. The first layer (in analogy to the sequence-to-structure network of earlier solutions [2, 3]) consisted of 32-filters each with a sliding window-size of w = 7. The second layer (structure-to-structure [2, 3]) created the final predictions by applying again a CNN (w = 7) over the output of the first layer. These two layers were connected through a rectified linear unit (ReLU) and a dropout layer [97] with a dropout-rate of 25% (Fig. 5, left panel). This simple architecture was trained independently on six different types of input, resulting in different number of free parameters. (i) DeepProf (14,000 = 14 k free parameters): Each residue was described by a vector of size 50 which included a one-hot encoding (20 features), the profiles of evolutionary information (20 features) from HHblits as published previously [46], the state transition probabilities of the Hidden-Markov-Model (7 features) and 3 features describing the local alignment diversity. (ii) DeepSeqVec (232 k free parameters): Each protein sequence was represented by the output of SeqVec. The

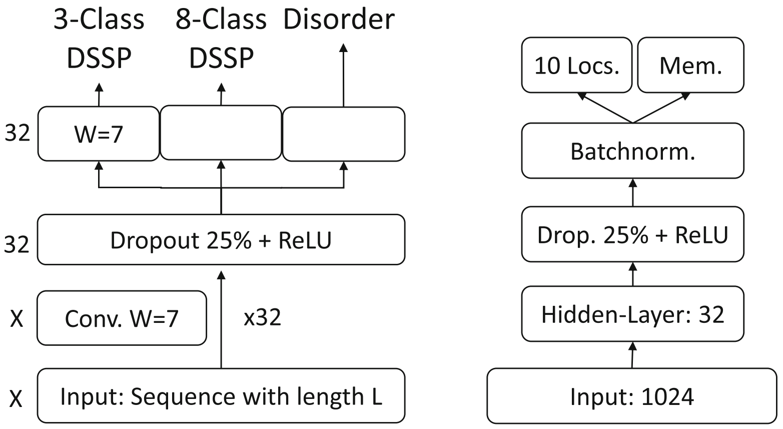

Fig. 5 Prediction tasks' architectures. On the left the architecture of the model used for the per-residue level predictions (secondary structure and disorder) is sketched, on the right that used for per-protein level predictions (localization and membrane/not membrane). The 'X', on the left, indicates that different input features corresponded to a difference in the number of input channels, e.g. 1024 for *SeqVec* or 50 for profile-based input. The letter 'W' refers to the window size of the corresponding convolutional layer (W = 7 implies a convolution of size 7 × 1)

resulting embedding described each residue as a 1024-dimensional vector. (iii) DeepProf+SeqVec (244 k free parameters): This model simply concatenated the input vectors used in (i) and (ii). (iv) DeepProtVec (25 k free parameters): Each sequence was split into overlapping 3-mers each represented by a 100-dimensional ProtVec [42]. (v) DeepOneHot (7 k free parameters): The 20 amino acids were encoded as one-hot vectors as described above. Rare amino acids were mapped to vectors with all components set to 0. Consequently, each protein residue was encoded as a 20-dimensional one-hot vector. (vi) DeepBLOSUM65 (8 k free parameters): Each protein residue was encoded by its BLOSUM65 substitution matrix [98]. In addition to the 20 standard amino acids, BLOSUM65 also contains substitution scores for the special cases B, Z (ambiguous) and X (unknown), resulting in a feature vector of length 23 for each residue.

On the per-protein level, a simple feed-forward neural network was used to demonstrate the power of the new embeddings. In order to ensure equal-sized input vectors for all proteins, we averaged over the 1024-dimensional embeddings of all residues in a given protein resulting in a 1024-dimensional vector representing any protein in the data set. ProtVec representations were derived the same way, resulting in a 100-dimensional vector. These vectors (either 100-or 1024 dimensional) were first compressed to 32 features, then dropout with a dropout rate of 25%, batch normalization [99] and a rectified linear Unit (ReLU) were applied before the final prediction (Fig. 5, right panel). In the following, we refer to the models trained on the two different input types as (i) DeepSeqVec-Loc (33 k free parameters): average over SeqVec embedding of a protein as described above and (ii) DeepProtVec-Loc (320 free parameters): average over ProtVec embedding of a protein. We used the following hyper-parameters: learning rate: 0.001, Adam optimizer with cross-entropy loss, batch size: 64. The losses of the individual tasks were summed before backpropagation. Due to the relatively small number of free parameters in our models, the training of all networks completed on a single Nvidia GeForce GTX1080 within a few minutes (11 s for DeepProtVec-Loc, 15 min for DeepSeqVec).

Evaluation measures

To simplify comparisons, we ported the evaluation measures from the publications we derived our data sets from, i.e. those used to develop *NetSurfP-2.0* [46] and *DeepLoc* [47]. All numbers reported constituted averages over all proteins in the final test sets. This work aimed at a proof-of-principle that the *SeqVec* embedding contain predictive information. In the absence of any claim for state-of-the-art performance, we did not calculate any significance values for the reported values.

Per-residue performance: Toward this end, we used the standard three-state per-residue accuracy (Q3 = percentage correctly predicted in either helix, strand, other [2]) along with its eight-state analog (Q8). Predictions of intrinsic disorder were evaluated through the Matthew's correlation coefficient (MCC [100]) and the False-Positive Rate (FPR) as those are more informative for tasks with high class imbalance. For completeness, we also provided the entire confusion matrices for both secondary structure prediction problems (Additional file 1: Figure S2). Standard errors were calculated over the distribution of each performance measure for all proteins.

Per-protein performance: The predictions whether a protein was membrane-bound or water-soluble were evaluated by calculating the two-state per set accuracy (Q2: percentage of proteins correctly predicted), and the MCC. A generalized MCC using the Gorodkin measure [101] for K (=10) categories as well as accuracy (Q10), was used to evaluate localization predictions. Standard errors were calculated using 1000 bootstrap samples, each chosen randomly by selecting a sub-set of the predicted test set that had the same size (draw with replacement).

Supplementary information

Supplementary information accompanies this paper at https://doi.org/10.1186/s12859-019-3220-8.

Additional file 1: Supporting online material (SOM) for: Modeling aspect of the language of life through transfer-learning protein sequences **Figure 1.** ELMo perplexity **Figure 2.** Confusion matrices for per-protein predictions using DeepSeqVec-Loc **Figure 3.** Confusion matrices for secondary structure predictions of DeepSeqVec **Figure 4.** Comparison of secondary structure prediction performance (Q3) between Netsurfp-2.0 and DeepSeqVec **Table S1.** Amino acid occurrences in UniRef50

Abbreviations

1D: One-dimensional – information representable in a string such as secondary structure or solvent accessibility; 3D structure: Three-dimensional coordinates of protein structure; 3D: Three-dimensional; ELMo: Embeddings from Language Models; MCC: Matthews-Correlation-Coefficient; MSA: Multiple sequence alignment; ProtVec: Context-independent embeddings from Word2vec-type approaches; Q10: Ten-state localization per-protein accuracy; Q3: Three-state secondary structure per-residue accuracy; Q8: Eight-state secondary structure per-residue accuracy; RSA: Relative solvent accessibility; SE: Standard error; SeqVec: embeddings introduced here, extracted by modeling un-annotated UniRef50 protein sequences with ELMo

Acknowledgements

The authors thank primarily Tim Karl for invaluable help with hardware and software and Inga Weise for support with many other aspects of this work. Last, not least, thanks to all those who deposit their experimental data in public databases, and to those who maintain these databases.

Authors contributions

AE and MH suggested to use ELMo for modeling protein sequences. AE adopted and trained ELMo. MH evaluated SeqVec embeddings on different data sets and tasks. YW helped with discussions about natural language processing. CD implemented the web-interface which allows to access and visualize the predictions and helped to improve the manuscript. DN helped with various problems regarding the code. FM and BR helped with the design of the experiment and to critically improve the manuscript. MH and AE drafted the manuscript and the other authors provided feedback. All authors read and approved the final manuscript.

Funding

This work was supported by a grant from the Alexander von Humboldt foundation through the German Ministry for Research and Education (BMBF: Bundesministerium fuer Bildung und Forschung) as well as by a grant from Deutsche Forschungsgemeinschaft (DFG–GZ: RO1320/4–1). We gratefully acknowledge the support of NVIDIA Corporation with the donation of two Titan GPU used for this research. We also want to thank the LRZ (Leibniz Rechenzentrum) for providing us access to DGX-V1.
The funding did not play any role in the design of the study, collection, analysis, and interpretation of data and in writing the manuscript.

Availability of data and materials

The pre-trained ELMo-based SeqVec model and a description on how to implement the embeddings into existing methods can be found here: https://github.com/Rostlab/SeqVec . Accessed 2nd May 2019.
Predictions on secondary structure, disorder and subcellular localization based on SeqVec can be accessed under: https://embed.protein.properties . Accessed 2nd May 2019.
The NetSurfP-2.0 data set [46] used for the evaluation of SeqVec on the task of secondary structure and disorder prediction are publicly available under: http://www.cbs.dtu.dk/services/NetSurfP/ . Accessed 2nd May 2019.
The DeepLoc data set [47] used for the evaluation of SeqVec on the task of subcellular localization prediction are publicly available under: http://www.cbs.dtu.dk/services/DeepLoc/data.php . Accessed 2nd May 2019.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Informatics, Bioinformatics & Computational Biology - i12, TUM (Technical University of Munich), Boltzmannstr. 3, 85748 Garching/Munich, Germany. [2]TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany. [3]Leibniz Supercomputing Centre, Boltzmannstr. 1, 85748 Garching/Munich, Germany. [4]TUM Department of Informatics, Software Engineering and Business Information Systems, Boltzmannstr. 1, 85748 Garching/Munich, Germany. [5]Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, 85748 Garching/Munich, Germany. [6]TUM School of Life Sciences Weihenstephan (WZW), Alte Akademie 8, Freising, Germany. [7]Department of Biochemistry and Molecular Biophysics & New York Consortium on Membrane Protein Structure (NYCOMPS), Columbia University, 701 West, 168th Street, New York, NY 10032, USA.

Received: 3 May 2019 Accepted: 13 November 2019
Published online: 17 December 2019

References

1. Rost B, Sander C. Jury returns on structure prediction. Nat. 1992;360:540.
2. Rost B, Sander C. Prediction of protein secondary structure at better than 70% accuracy. J Mol Biol. 1993;232:584–99.
3. Rost B, Sander C. Improved prediction of protein secondary structure by use of sequence profiles and neural networks. Proc Natl Acad Sci. 1993;90:7558–62.
4. Barton GJ. Protein secondary structure prediction. Curr Opin Struct Biol. 1995;5:372–6.
5. Chandonia J-M, Karplus M. Neural networks for secondary structure and structural class predictions. Protein Sci. 1995;4:275–85.
6. Mehta PK, Heringa J, Argos P. A simple and fast approach to prediction of protein secondary structure from multiply aligned sequences with accuracy above 70%. Protein Sci. 1995;4:2517–25.
7. Rost B, Sander C. Combining evolutionary information and neural networks to predict protein secondary structure. Proteins Struct Funct Genet. 1994;19:55–72.
8. Solovyev VV, Salamov AA. Predicting a-helix and b-strand segments of globular proteins. Comput Appl Biol Sci. 1994;10:661–9.
9. Frishman D, Argos P. Knowledge-based protein secondary structure assignment. Proteins Struct Funct Genet. 1995;23:566–79.
10. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. J Mol Biol. 1999;292(2):195–202.
11. Bigelow H, Petrey D, Liu J, Przybylski D, Rost B. Predicting transmembrane beta-barrels in proteomes. Nucleic Acids Res. 2004;32:2566–77.
12. Rost B, Casadio R, Fariselli P. Topology prediction for helical transmembrane proteins at 86% accuracy. Protein Sci. 1996;5:1704–18.
13. Rost B, Casadio R, Fariselli P, Sander C. Transmembrane helix prediction at 95% accuracy. Protein Sci. 1995;4:521–33.

14. Rost B, Sander C. Conservation and prediction of solvent accessibility in protein families. Proteins Struct Funct Genet. 1994;20(3):216–26.
15. Radivojac P, Obradovic Z, Smith DK, Zhu G, Vucetic S, Brown CJ, Lawson JD, Dunker AK. Protein flexibility and intrinsic disorder. Protein Sci. 2004;13:71–80.
16. Schlessinger A, Rost B. Protein flexibility and rigidity predicted from sequence. Proteins. 2005;61(1):115–26.
17. Punta M, Rost B. PROFcon: novel prediction of long-range contacts. Bioinform. 2005;21(13):2960–8.
18. Peng K, Vucetic S, Radivojac P, Brown CJ, Dunker AK, Obradovic Z. Optimizing long intrinsic disorder predictors with protein evolutionary information. J Bioinforma Comput Biol. 2005;3(1):35–60.
19. Schlessinger A, Liu J, Rost B. Natively unstructured loops differ from other loops. PLoS Comput Biol. 2007;3(7):e140.
20. Schlessinger A, Punta M, Rost B. Natively unstructured regions in proteins identified from contact predictions. Bioinform. 2007;23(18):2376–84.
21. Nair R, Rost B. Better prediction of sub-cellular localization by combining evolutionary and structural information. Proteins. 2003;53(4):917–30.
22. Nair R, Rost B. Mimicking cellular sorting improves prediction of subcellular localization. J Mol Biol. 2005;348(1):85–100.
23. Marino Buslje C, Teppa E, Di Domenico T, Delfino JM, Nielsen M. Networks of high mutual information define the structural proximity of catalytic sites: implications for catalytic residue identification. PLoS Comput Biol. 2010; 6(11):e1000978.
24. Ofran Y, Rost B. Protein-protein interaction hot spots carved into sequences. PLoS Comput Biol. 2007;3(7):e119.
25. Ofran Y, Rost B. ISIS: interaction sites identified from sequence. Bioinform. 2007;23(2):e13–6.
26. Adzhubei IA, Schmidt S, Peshkin L, Ramensky VE, Gerasimova A, Bork P, Kondrashov AS, Sunyaev SR. A method and server for predicting damaging missense mutations. Nat Methods. 2010;7(4):248–9.
27. Bromberg Y, Rost B. SNAP: predict effect of non-synonymous polymorphisms on function. Nucleic Acids Res. 2007;35(11):3823–35.
28. Hayat S, Sander C, Marks DS, Elofsson A. All-atom 3D structure prediction of transmembrane β-barrel proteins from sequences. Proc Natl Acad Sci. 2015; 112(17):5413–8.
29. Marks DS, Colwell LJ, Sheridan R, Hopf TA, Pagnani A, Zecchina R, Sander C. Protein 3D structure computed from evolutionary sequence variation. PLoS One. 2011;6(12):e28766.
30. Marks DS, Hopf TA, Sander C. Protein structure prediction from sequence variation. Nat Biotechnol. 2012;30(11):1072.
31. Morcos F, Pagnani A, Lunt B, Bertolino A, Marks DS, Sander C, Zecchina R, Onuchic JN, Hwa T, Weigt M. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. Proc Natl Acad Sci. 2011;108(49):E1293–301.
32. Suzek BE, Wang Y, Huang H, McGarvey PB, Wu CH, UniProt C. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. Bioinform. 2015;31(6):926–32.
33. Altschul SF, Madden TL, Schaeffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped Blast and PSI-Blast: a new generation of protein database search programs. Nucleic Acids Res. 1997;25:3389–402.
34. Remmert M, Biegert A, Hauser A, Soding J. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. Nat Methods. 2012;9(2):173–5.
35. Steinegger M, Meier M, Mirdita M, Vohringer H, Haunsberger SJ, Soding J. HH-suite3 for fast remote homology detection and deep protein annotation. BMC Bioinform. 2019;20(1):473.
36. Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nat Biotechnol. 2017; 35(11):1026.
37. Dunker AK, Babu MM, Barbar E, Blackledge M, Bondos SE, Dosztanyi Z, Dyson HJ, Forman-Kay J, Fuxreiter M, Gsponer J, et al. What's in a name? Why these proteins are intrinsically disordered. Intrinsically Disord Proteins. 2013;1(1):e24157.
38. Uversky VN, Radivojac P, Iakoucheva LM, Obradovic Z, Dunker AK. Prediction of intrinsic disorder and its use in functional proteomics. Methods Mol Biol. 2007;408:69–92.
39. Perdigao N, Heinrich J, Stolte C, Sabir KS, Buckley MJ, Tabor B, Signal B, Gloss BS, Hammang CJ, Rost B, et al. Unexpected features of the dark proteome. Proc Natl Acad Sci U S A. 2015.
40. Schafferhans A, O'Donoghue SI, Heinzinger M, Rost B. Dark proteins important for cellular function. Proteomics. 2018;18(21–22):1800227.
41. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L: Deep contextualized word representations. arXiv 2018,.https://arxiv.org/abs/1802.05365.
42. Asgari E, Mofrad MR. Continuous distributed representation of biological sequences for deep proteomics and genomics. PLoS One. 2015;10(11):e0141287.
43. Mikolov T, Chen K, Corrado G, Dean J: Efficient estimation of word representations in vector space. ArXiv 2013,https://arxiv.org/abs/1301.3781.
44. Schils E, Pd H. Characteristics of sentence length in running text. Literary Linguist Comput. 1993;8(1):20–6.
45. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
46. Klausen MS, Jespersen MC, Nielsen H, Jensen KK, Jurtz VI, Sonderby CK, Sommer MOA, Winther O, Nielsen M, Petersen B, et al. NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning. Proteins. 2019.
47. Almagro Armenteros JJ, Sonderby CK, Sonderby SK, Nielsen H, Winther O. DeepLoc: prediction of protein subcellular localization using deep learning. Bioinform. 2017;33(24):4049.
48. Anfinsen CB. Principles that govern the folding of protein chains. Sci. 1973; 181(4096):223–30.
49. Buchan DW, Jones DT. Improved protein contact predictions with the MetaPSICOV2 server in CASP12. Proteins. 2018;86:78–83.
50. Evans R, Jumper J, Kirkpatrick J, Sifre L, Green T, Qin C, Zidek A, Nelson A, Bridgland A, Penedones H. De novo structure prediction with deeplearning based scoring. Annu Rev Biochem. 2018;77(363–382):6.
51. Rives A, Goyal S, Meier J, Guo D, Ott M, Zitnick CL, Ma J, Fergus R. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. bioRxiv. 2019:622803.
52. Chou KC, Wu ZC, Xiao X. iLoc-Euk: a multi-label classifier for predicting the subcellular localization of singleplex and multiplex eukaryotic proteins. PLoS One. 2011;6(3):e18258.
53. Lvd M, Hinton G. Visualizing data using t-SNE. J Mach Learn Res. 2008; 9(Nov):2579–605.
54. Fox NK, Brenner SE, Chandonia J-M. SCOPe: structural classification of proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. Nucleic Acids Res. 2013;42(D1):D304–9.
55. Kosloff M, Kolodny R. Sequence-similar, structure-dissimilar protein pairs in the PDB. Proteins. 2008;71(2):891–902.
56. Dai Z, Yang Z, Yang Y, Cohen WW, Carbonell J, Le QV, Salakhutdinov R: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:190102860 2019.
57. Devlin J, Chang M-W, Lee K, Toutanova K: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv: 181004805 2018.
58. Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov R, Le QV: XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv preprint arXiv:190608237 2019.
59. AlQuraishi M. ProteinNet: a standardized data set for machine learning of protein structure. BMC Bioinform. 2019;20(1):311.
60. Bairoch A. The ENZYME database in 2000. Nucleic Acids Res. 2000;28(1):304–5.
61. Velankar S, Dana JM, Jacobsen J, Van Ginkel G, Gane PJ, Luo J, Oldfield TJ, O'donovan C, Martin M-J, Kleywegt GJ. SIFTS: structure integration with function, taxonomy and sequences resource. Nucleic Acids Res. 2012;41(D1):D483–9.
62. Heffernan R, Yang Y, Paliwal K, Zhou Y. Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility. Bioinform. 2017;33(18):2842–9.
63. Wang S, Li W, Liu S, Xu J. RaptorX-property: a web server for protein structure property prediction. Nucleic Acids Res. 2016;44(W1):W430–5.
64. Wang S, Peng J, Ma J, Xu J. Protein secondary structure prediction using deep convolutional neural fields. Sci Rep. 2016;6:18962.
65. Drozdetskiy A, Cole C, Procter J, Barton GJ. JPred4: a protein secondary structure prediction server. Nucleic Acids Res. 2015;43(W1):W389–94.
66. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE. The protein data bank. Nucleic Acids Res. 2000; 28(1):235–42.
67. Wang G, Dunbrack RL Jr. PISCES: a protein sequence culling server. Bioinform. 2003;19(12):1589–91.
68. Kabsch W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen bonded and geometrical features. Biopolym. 1983; 22:2577–637.

69. Yang Y, Gao J, Wang J, Heffernan R, Hanson J, Paliwal K, Zhou Y. Sixty-five years of the long march in protein secondary structure prediction: the final stretch? Brief Bioinform. 2016;19(3):482–94.
70. Cuff JA, Barton GJ. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. Proteins Struct Funct Genet. 1999;34(4):508–19.
71. Abriata LA, Tamò GE, Monastyrskyy B, Kryshtafovych A, Dal Peraro M. Assessment of hard target modeling in CASP12 reveals an emerging role of alignment-based contact prediction methods. Proteins. 2018;86:97–112.
72. Goldberg T, Hamp T, Rost B. LocTree2 predicts localization for all domains of life. Bioinform. 2012;28(18):i458–65.
73. Blum T, Briesemeister S, Kohlbacher O. MultiLoc2: integrating phylogeny and gene ontology terms improves subcellular protein localization prediction. BMC Bioinform. 2009;10:274.
74. Briesemeister S, Blum T, Brady S, Lam Y, Kohlbacher O, Shatkay H. SherLoc2: a high-accuracy hybrid method for predicting subcellular localization of proteins. J Proteome Res. 2009;8(11):5363–6.
75. Yu CS, Chen YC, Lu CH, Hwang JK. Prediction of protein subcellular localization. Proteins. 2006;64(3):643–51.
76. Horton P, Park KJ, Obayashi T, Fujita N, Harada H, Adams-Collier CJ, Nakai K. WoLF PSORT: protein localization predictor. Nucleic Acids Res. 2007;35(Web Server issue):W585–7.
77. Briesemeister S, Rahnenfuhrer J, Kohlbacher O. YLoc - an interpretable web server for predicting subcellular localization. Nucleic Acids Res. 2010; 38(Suppl):W497–502.
78. Boutet E, Lieberherr D, Tognolli M, Schneider M, Bansal P, Bridge AJ, Poux S, Bougueleret L, Xenarios I. UniProtKB/Swiss-Prot, the manually annotated section of the UniProt KnowledgeBase: how to use the entry view. Methods Mol Biol. 2016;1374:23–54.
79. Fu L, Niu B, Zhu Z, Wu S, Li W. CD-HIT: accelerated for clustering the next-generation sequencing data. Bioinform. 2012;28(23):3150–2.
80. Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinform. 2006;22(13):1658–9.
81. Moussa M, Mandoiu II. Single cell RNA-seq data clustering using TF-IDF based methods. BMC Genomics. 2018;19(Suppl 6):569.
82. Bailey TL, Boden M, Buske FA, Frith M, Grant CE, Clementi L, Ren J, Li WW, Noble WS. MEME SUITE: tools for motif discovery and searching. Nucleic Acids Res. 2009;37(Web Server issue):W202–8.
83. Bernard G, Chan CX, Ragan MA. Alignment-free microbial phylogenomics under scenarios of sequence divergence, genome rearrangement and lateral genetic transfer. Sci Rep. 2016;6:28970.
84. Hamp T, Rost B. Evolutionary profiles improve protein-protein interaction prediction from sequence. Bioinform. 2015;31(12):1945–50.
85. Kuang R, Ie E, Wang K, Wang K, Siddiqi M, Freund Y, Leslie C. Profile-based string kernels for remote homology detection and motif extraction. J Bioinforma Comput Biol. 2005;3(3):527–50.
86. Leslie C, Eskin E, Weston J, Noble WS: Mismatch string kernels for SVM protein classification. Bioinform 2003:in press.
87. Nakai K, Horton P. PSORT: a program for detecting sorting signals in proteins and predicting their subcellular localization. Trends Biochem Sci. 1999;24(1):34–6.
88. Noble WS, Kuang R, Leslie C, Weston J. Identifying remote protein homologs by network propagation. FEBS J. 2005;272(20):5119–28.
89. Asgari E, McHardy AC, Mofrad MRK. Probabilistic variable-length segmentation of protein sequences for discriminative motif discovery (DiMotif) and sequence embedding (ProtVecX). Sci Rep. 2019;9(1):3577.
90. Kim S, Lee H, Kim K, Kang J. Mut2Vec: distributed representation of cancerous mutations. BMC Med Genet. 2018;11(2):33.
91. Xu Y, Song J, Wilson C, Whisstock JC. PhosContext2vec: a distributed representation of residue-level sequence contexts and its application to general and kinase-specific phosphorylation site prediction. Sci Rep. 2018;8.
92. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. Trans Assoc Comput Linguist. 2017;5:135–46.
93. Pennington J, Socher R, Manning C: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP): 2014. 1532–1543.
94. Kim Y, Jernite Y, Sontag D, Rush AM: Character-aware neural language models. In: Thirtieth AAAI Conference on Artificial Intelligence: 2016.
95. Reddi SJ, Kale S, Kumar S: On the convergence of adam and beyond. arXiv preprint arXiv:190409237 2019.
96. Kingma DP, Ba J: Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980 2014.
97. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
98. Henikoff S, Henikoff JG. Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci. 1992;89(22):10915–9.
99. Ioffe S, Szegedy C: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:150203167 2015.
100. Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta. 1975;405:442–51.
101. Gorodkin J. Comparing two K-category assignments by a K-category correlation coefficient. Comput Biol Chem. 2004;28(5–6):367–74.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

3. Embedding-based Annotation Transfer (EAT) for GO prediction - goPredSim

3.1. Preface

The Gene Ontology (GO) [108] makes protein function human- and machine-readable by mapping it to a structured and controlled vocabulary. In order to take the fuzzy meaning of function into account, GO provides three hierarchies each looking at different aspects of function: the Molecular Function Ontology (MFO) defines a protein's function at the molecular level, while the Biological Process Ontology (BPO) considers the larger context or pathway a protein acts in and the Cellular Component Ontology (CCO) defines the sub-cellular environment in which a protein functions in. Further, a protein can be assigned to either of the ontologies or all of them at varying levels of specificity depending on the experimental setup and its resolution. This also allows to catch a glimpse of the complexity arising from trying to determine a protein's function(s): There is no way to measure the vast space of protein functions put forth by nature over millions of years in an extensive way. Rather, there are single pieces of evidence that need to be put together to get a comprehensive view on a protein's function(s). On top, some protein's function(s) do not only crucially depend on their cellular environment, which might be hard to replicate experimentally but they might even alter their function based on their environment. This makes resolving a protein's function(s) extremely time-consuming and expensive and makes computational approaches attractive that try to bypass these problems by predicting GO terms.

We used SeqVec embeddings (Chapter 2) to develop a new and simple method, dubbed *goPredSim*, to predict GO terms from single protein sequences. Similar to homology-based inference (HBI) which transfers annotations from a set of labeled proteins to a set of unlabeled proteins, we also transferred annotations but instead of relying on sequence similarity, we used Euclidean distance between SeqVec embeddings to define similarity. Crucially, this approach is not limited to the prediction of GO terms but as generally applicable as HBI. To put the performance of this embedding-based annotation transfer (EAT) into perspective of other methods, we replicated the conditions of the third Critical Assessment of protein Function Annotation algorithms (CAFA3) which assesses computational methods that predict GO terms.

According to the CAFA3 benchmark [38], goPredSim would have reached Fmax = 37±2%, 50±3%, and 57±2% for BPO, MFO, and CCO, respectively, making it competitive to the top ten CAFA3 competitors if we had participated. This was supported by preliminary evaluations for CAFA4 presented at ISMB2020 [109]. That our extremely simple and generally applicable approach remained competitive to sophisticated machine learning approaches tailored for GO term prediction, highlights the effectiveness of our proposed protein sequence representations. On top, in direct comparison, EAT clearly outperformed HBI, indicating that embeddings capture functional similarity beyond sequence similarity. The method is available as GitHub repository (https://github.com/Rostlab/goPredSim), as part of the *PredictProtein* web-server [110] and under https://embed.protein.properties/ which relies on *bio_embeddings* [107].

Author contribution: I contributed to the original conceptualisation and provided various embeddings. Maria Littmann implemented the method, and performed the evaluation. Christian Dallago implemented the web server. Tobias Olenyi computed the combination of sequence and embedding-based annotation transfer. All authors drafted the manuscript.

3.2. Journal Article: Littmann, Heinzinger *et al.*, Scientific Reports (2021)

Reference: Littmann, M., Heinzinger, M., Dallago, C., Olenyi, T., and Rost, B. Embeddings from deep learning transfer GO annotations beyond homology. *Scientific Reports*, 11(1):2045–2322, 2021. doi:10.1038/s41598-020-80786-0

www.nature.com/scientificreports

scientific reports

Check for updates

OPEN

Embeddings from deep learning transfer GO annotations beyond homology

Maria Littmann[1,2,6]✉, Michael Heinzinger[1,2,6], Christian Dallago[1,2], Tobias Olenyi[1] & Burkhard Rost[1,3,4,5]

Knowing protein function is crucial to advance molecular and medical biology, yet experimental function annotations through the Gene Ontology (GO) exist for fewer than 0.5% of all known proteins. Computational methods bridge this sequence-annotation gap typically through homology-based annotation transfer by identifying sequence-similar proteins with known function or through prediction methods using evolutionary information. Here, we propose predicting GO terms through annotation transfer based on proximity of proteins in the SeqVec embedding rather than in sequence space. These embeddings originate from deep learned language models (LMs) for protein sequences (SeqVec) transferring the knowledge gained from predicting the next amino acid in 33 million protein sequences. Replicating the conditions of CAFA3, our method reaches an F_{max} of 37 ± 2%, 50 ± 3%, and 57 ± 2% for BPO, MFO, and CCO, respectively. Numerically, this appears close to the top ten CAFA3 methods. When restricting the annotation transfer to proteins with < 20% pairwise sequence identity to the query, performance drops (F_{max} BPO 33 ± 2%, MFO 43 ± 3%, CCO 53 ± 2%); this still outperforms naïve sequence-based transfer. Preliminary results from CAFA4 appear to confirm these findings. Overall, this new concept is likely to change the annotation of proteins, in particular for proteins from smaller families or proteins with intrinsically disordered regions.

Abbreviations

BERT	Bidirectional Encoder Representations from Transformers (particular deep learning language model)
BP(O)	Biological process (ontology) from GO
CAFA	Critical Assessment of Functional Annotation
CC(O)	Cellular component (ontology) from GO
ELMo	Embeddings from Language Models
GO	Gene ontology
GOA	Gene Ontology Annotation
k-NN	K-nearest neighbor
LK	Limited-knowledge
LM	Language model
LSTMs	Long-short-term-memory cells
M	Million
MF(O)	Molecular function (ontology) from GO
NK	No-knowledge
PIDE	Percentage pairwise sequence identity
RI	Reliability index
RMSD	Root-mean-square deviation

[1]Department of Informatics, Bioinformatics and Computational Biology, i12, TUM (Technical University of Munich), Boltzmannstr. 3, Garching, 85748 Munich, Germany. [2]TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany. [3]Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, Garching, 85748 Munich, Germany. [4]School of Life Sciences Weihenstephan (TUM-WZW), TUM (Technical University of Munich), Alte Akademie 8, Freising, Germany. [5]Department of Biochemistry and Molecular Biophysics, Columbia University, 701 West, 168th Street, New York, NY 10032, USA. [6]These authors contributed equally: Maria Littmann and Michael Heinzinger. ✉email: littmann@rostlab.org

GO captures cell function through hierarchical ontologies. All organisms rely on the correct functioning of their cellular workhorses, namely their proteins involved in almost all roles, ranging from molecular functions (MF) such as chemical catalysis of enzymes to biological processes or pathways (BP), e.g., realized through signal transduction. Only the perfectly orchestrated interplay between proteins allows cells to perform more complex functions, e.g., the aerobic production of energy via the citric acid cycle requires the interconnection of eight different enzymes with some of them being multi-enzyme complexes[1]. The Gene Ontology (GO)[2] thrives to capture this complexity and to standardize the vocabulary used to describe protein function in a human- and machine-readable manner. GO separates different aspects of function into three hierarchies: MFO (Molecular Function Ontology), BPO (biological process ontology), and CCO, i.e. the cellular component(s) or subcellular localization(s) in which the protein acts.

Computational methods bridge the sequence-annotation gap. As the experimental determination of complete GO numbers is challenging, the gap between the number of proteins with experimentally verified GO numbers and those with known sequence but unknown function (sequence-annotation gap) remains substantial. For instance, UniRef100[3] (UniProt[3] clustered at 100% percentage pairwise sequence identity, PIDE) contains roughly 220 M (million) protein sequences of which fewer than 1 M have annotations verified by experts (Swiss-Prot[3] evidence codes EXP, IDA, IPI, IMP, IGI, IEP, TAS, or IC).

Computational biology has been bridging the sequence-annotation gap for decades[4–11], based on two different concepts: (1) sequence similarity-based transfer (or *homology-based inference*) which copies the annotation from one protein to another if that is sequence similar enough because proteins of similar sequence have similar function[12]. In more formal terms: given a query Q of unknown and a template T of known function (F_t): IF PIDE(Q,T) > threshold θ, transfer annotation F_t to Q. (2) *De-novo* methods predict protein function through machine learning[5]. If applicable, the first approach tends to out-perform the second[13–16] although it largely misses discoveries[17]. The progress of computational methods has been monitored by CAFA (*Critical Assessment of Functional Annotation*)[9,18,19], an international collaboration for advancing and assessing methods that bridge the sequence-annotation gap. CAFA takes place every 2–3 years with its fourth instance (CAFA4) currently being evaluated.

Here, we introduce a novel approach transferring annotations using the similarity of embeddings from language models (LMs: SeqVec[20] and ProtBert[21]) rather than the similarity of sequence. Using embedding space proximity has helped information retrieval in natural language processing (NLP)[22–25]. By learning to predict the next amino acid given the entire previous sequence on unlabeled data (only sequences without any phenotype/label), e.g., SeqVec learned to extract features describing proteins useful as input to different tasks (transfer learning). Instead of transferring annotations from the labeled protein T with the highest percentage pairwise sequence identity (PIDE) to the query Q, we chose T as the protein with the smallest distance in embedding space ($DIST^{emb}$) to Q. This distance also proxied the reliability of the prediction serving as threshold above which hits are considered too distant to infer annotations. Instead of picking the top hit, annotations can be inferred from the *k* closest proteins where *k* has to be optimized. In addition, we evaluate the influence of the type of LM used (SeqVec[20] vs. ProtBert[21]). Although the LMs were never trained on GO terms, we hypothesize LM embeddings to implicitly encode information relevant for the transfer of annotations, i.e., capturing aspects of protein function because embeddings have been shown to capture rudimentary features of protein structure and function[20,21,26,27].

Results and discussion

Simple embedding-based transfer almost as good as CAFA3 top-10. First, we predicted GO terms for all 3328 CAFA3 targets using the Gene Ontology Annotation (GOA) data set *GOA2017* (Methods), removed all entries identical to CAFA3 targets (PIDE = 100%; set: *GOA2017-100*) and transferred the annotations of the closest hit (k = 1; closest by Euclidean distance) in this set to the query. When applying the *NK evaluation mode* (no-knowledge available for query, Methods/CAFA3), the embedding-transfer reached F_{max} scores (Eq. 3) of 37 ± 2% for BPO (precision: P = 39 ± 2%, recall: R = 36 ± 2%, Eqs. 1/2), F1 = 50 ± 3% for MFO (P = 54 ± 3%, R = 47 ± 3%), and F1 = 57 ± 2% for CCO (P = 61 ± 3%, R = 54 ± 3%; Table 1, Fig. 1, Fig. S1). Errors were estimated through the 95% confidence intervals (± 1.96 stderr). Replacing the Euclidean by cosine distance (more standard amongst those working with embeddings, e.g., in NLP) changed little (Table 1; for simplicity, we only used Euclidean from here on). In the sense that the database with annotations to transfer (GOA2017) had been available before the CAFA3 submission deadline (February 2017), our predictions were directly comparable to CAFA3[19]. This embedding-based annotation transfer clearly outperformed the two CAFA3 baselines (Fig. 1: the simple *BLAST* for sequence-based annotation transfer, and the *Naïve* method assigning GO terms statistically based on database frequencies, here *GOA2017*); it would have performed close to the top ten CAFA3 competitors (in particular for BPO: Fig. 1) had the method competed at CAFA3.

Performance did not change when replacing global average with maximum pooling (Table 1). While averaging over long proteins could lead to information loss in the resulting embeddings, we did not observe a correlation between performance and protein length (Fig. S2, Table S1). In order to obtain the embeddings, we processed query and lookup protein the same way. If those have similar function and similar length, their embeddings might have lost information in the same way. This loss might have "averaged out" to generate similar embeddings.

Including more neighbors ($k > 1$) only slightly affected F_{max} (Table S2; all F_{max} averages for $k = 2$ to $k = 10$ remained within the 95% confidence interval of that for $k = 1$). When taking all predictions into account independent of a threshold in prediction strength referred to as the reliability index (RI, Methods; i.e., even low confidence annotations are transferred), the number of predicted GO terms increased with higher *k* (Table S3). The average number of GO terms annotated per protein in *GOA2017* already reached 37, 14, 9 for BPO, MFO, CCO, respectively. When including all predictions independent of their strength (RI) our method predicted more

Data set	*Embeddings*	*F_{max}*		
		BPO	*MFO*	*CCO*
GOA2017	*SeqVec*	37 ± 2%	50 ± 3%	57 ± 2%
	SeqVec (Cosine)	37 ± 2%	50 ± 3%	58 ± 2%
	SeqVec (maximum pooling)	35 ± 2%	52 ± 3%	58 ± 2%
	ProtBert	36 ± 2%	49 ± 3%	59 ± 2%
	BLAST	26%	42%	46%
GOA2017X	*SeqVec*	31 ± 2%	51 ± 3%	56 ± 2%
GOA2020	*SeqVec*	51 ± 2%	61 ± 3%	65 ± 2%
	ProtBert	50 ± 2%	59 ± 2%	65 ± 2%
	BLAST	31%	53%	58%

Table 1. Performance for CAFA3 targets for simple GO annotation transfers*. *Mean F_{max} values for GO term predictions using embeddings from two different language models (*SeqVec* or *ProtBert*) or sequence similarity (*BLAST*) for the data sets *GOA2017-100* (2017), *GOA2017X* (2017), and *GOA2020-100* (2020) used for annotation transfer (note: the notation '-100′ implies that any entry in the data set with PIDE = 100% to any CAFA3 protein had been removed). By default, embedding distance was assessed by Euclidean distance (Eq. 4; exception marked *cosine*), and per-residue embeddings were pooled by global average pooling (exception marked maximum pooling). All values were compiled for picking the single top hit (k = 1) and using the CAFA3 targets from the NK and full evaluation mode[19]. For all simple annotation transfers (embedding- and sequence-based), performance was higher for the more recent data sets (*GOA2020* vs. *GOA2017*). Error estimates are given as 95% confidence intervals. F_{max} values were computed using the CAFA3 tool[18,19].

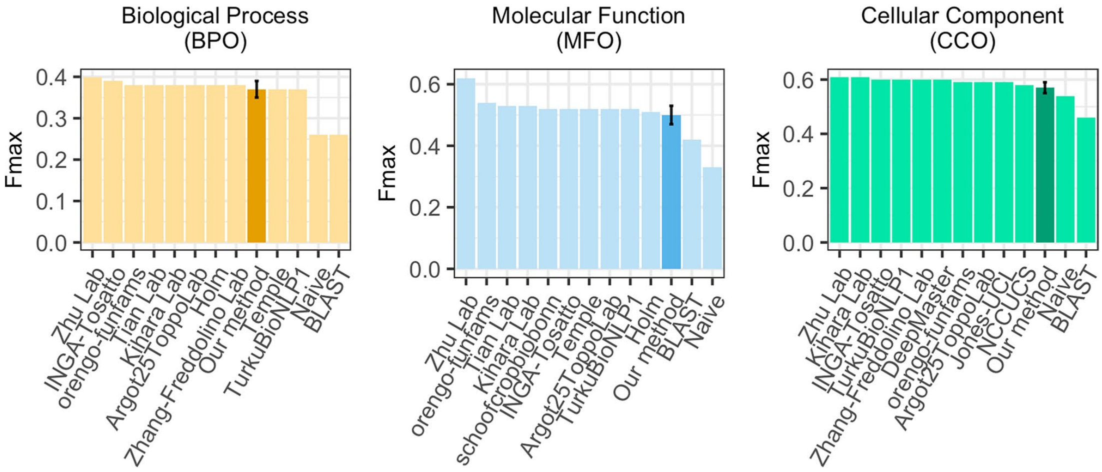

Figure 1. **F_{max} for simplest embedding-based transfer (k = 1) and CAFA3 competitors.** Using the data sets and conditions from CAFA3, we compared the F_{max} of the simplest implementation of the embedding-based annotation transfer, namely the greedy (k = 1) solution in which the transfer comes from exactly one closest database protein (dark bar) for the three ontologies (BPO, MFO, CCO) to the top ten methods that—in contrast to our method—did compete at CAFA3 and to two background approaches "BLAST" (homology-based inference) and "Naïve" (assignment of terms based on term frequency) (lighter bars). The result shown holds for the *NK* evaluation mode (no knowledge), i.e., only using proteins that were novel in the sense that they had no prior annotations. If we had developed our method before CAFA3, it would have almost reached the tenth place for MFO and CCO, and ranked even slightly better for BPO. Error bars (for our method) marked the 95% confidence intervals.

terms for CCO and BPO than expected from this distribution even for $k = 1$. Only for MFO the average (11.7 terms) predicted was slightly lower than expected for $k = 1$ (number of terms exploded for $k > 1$: Table S3). While precision dropped with adding terms, recall increased (Table S3). To avoid overprediction and given that k hardly affected F_{max}, we chose $k = 1$. This choice might not be best in practice: considering more than one hit ($k > 1$) might help when the closest hit only contains unspecific terms. However, such a distinction will be left to expert users.

When applying the LK evaluation mode (limited-knowledge, i.e., query already has some annotation about function, Methods/CAFA3), the embedding-based annotation transfer reached F_{max} scores of 49 ± 1%, 52 ± 2%,

	PIDE	d_{SeqVec}	$d_{ProtBert}$
PIDE	1	0.293	0.248
d_{SeqVec}		1	0.576
$d_{ProtBert}$			1

Table 2. Embedding and sequence similarity correlated*. ***PIDE* percentage pairwise sequence identity, d_{SeqVec} similarity in SeqVec[20] embeddings (Eq. 5); $d_{ProtBert}$ similarity in ProtBert[21] embeddings (Eq. 5). All values represent Spearman's correlation coefficients calculated for 434,001 sequence pairs. For all pairs, the significance was *p*-value < 2.2e−16, i.e., significant at the level of the precision of the software R[28]. The similarity between sequence and embeddings correlated less than the two different types of embeddings, namely SeqVec and ProtBert with each other. In order to highlight the trivial symmetry of the matrix, only the upper diagonal was given.

and 58 ± 3% for BPO, MFO, and CCO, respectively (Fig. S3). Thus, the embedding-based annotation transfer reached higher values for proteins with prior annotations (LK evaluation mode) than for novel proteins without any annotations (NK evaluation mode; Table 1); the same was true for the CAFA3 top-10 for which the F_{max} scores increased even more than for our method for BPO and MFO, and less for CCO (Fig. 1, Fig. S3). In the *LK* mode, predictions are evaluated for proteins for which 1–2 GO ontologies had annotations while those for another ontology (or two) were added after the CAFA3 deadline[9,19]. While supervised training uses such labels; our approach did not since we had excluded all CAFA3 targets explicitly from the annotation transfer database (*GOA2017*). Thus, our method could not benefit from previous annotations, i.e., *LK* and *NK* should be identical. The observed differences were most likely explained by how F_{max} is computed. The higher F_{max} score, especially for BPO, might be explained by data set differences, e.g. LK average number of BPO annotations was 19 compared to 26 for NK. Other methods might have reached even higher by training on known annotations.

Embedding-based transfer successfully identified distant relatives.

Embeddings condense information learned from sequences; identical sequences produce identical embeddings: if PIDE(Q,T) = 100%, then $DIST^{emb}(Q,T) = 0$ (Eq. 4). We had assumed a simple relation: the more similar two sequences, the more similar their embeddings because the underlying LMs only use sequences as input. Nevertheless, we observed embedding-based annotation transfer to outperform (higher F_{max}) sequence-based transfer (Table 1, Fig. 1). This suggested embeddings to capture information beyond raw sequences. Explicitly calculating the correlation between sequence and embedding similarity for 431,224 sequence pairs from CAFA3/*GOA2017*-100, we observed a correlation of $\rho = 0.29$ (Spearman's correlation coefficient, *p*-value < 2.2e−16; Table 2). Thus, sequence and embedding similarity correlated at an unexpectedly low level. However, our results demonstrated that embedding similarity identified more distant relatives than sequence similarity (Figs. S1, S4).

In order to quantify how different embeddings for proteins Q and T can still share GO terms, we redundancy reduced the *GOA2017* database used for annotation transfers at distance thresholds of decreasing PIDE with respect to the queries (in nine steps from 90 to 20%, Table S6). By construction, all proteins in *GOA2017-100* had PIDE < 100% to all CAFA3 queries (Q). If the pair (Q,T) with the most similar embedding was also similar in sequence, embedding-based would equal sequence-based transfer. At lower PIDE thresholds, e.g., PIDE < 20%, reliable annotation transfer through simple pairwise sequence alignment is no longer possible[14,29–32]. Although embeddings-based transfer tended to be slightly less successful for pairs with lower PIDE (Fig. 2: bars decrease toward right), the drop appeared small; on top, at almost all PIDE values, embedding-transfer remained above BLAST, i.e., sequence-based transfer (Fig. 2: most bars higher than reddish line – error bars show 95% confidence intervals). The exception was for MFO at PIDE < 30% and PIDE < 20% for which the F_{max} scores from sequence-based transfer (BLAST) were within the 95% confidence interval (Fig. 2). This clearly showed that our approach benefited from information available through embeddings but not through sequences, and that at least some protein pairs close in embedding and distant in sequence space might function alike. In order to correctly predict the next token, protein LMs have to learn complex correlations between residues as it is impossible to remember the multitude of all possible amino acid combinations in hundreds of millions to billions of protein sequences. This forces models to abstract higher level features from sequences. For instance, secondary structure can directly be extracted from embeddings through linear projection[26]. The LMs (SeqVec & ProtBert) might even have learned to find correlations between protein pairs diverged into the "midnight zone" sequence comparison in which sequence similarity becomes random[29,33]. Those cases are especially difficult to detect by the most successful search methods such as BLAST[34] or MMseqs2[35] relying on finding similar seeds missing at such diverged levels.

Ultimately, we failed to really explain why the abstracted level of sequences captured in embeddings outperformed raw sequences. One attempt at addressing this question led to displaying cases for which one of the two worked better (Fig. S5). Looking in more detail at outliers (embeddings more similar than sequences), we observed that embedding-based inference tended to identify more reasonable hits in terms of lineage or structure. For instance, for the uncharacterized transporter YIL166C (UniProt identifier P40445) from *Saccharomyces cerevisiae* (baker's yeast), the closest hit in SeqVec embedding space was the high-affinity nicotinic acid transporter (P53322) also from *Saccharomyces cerevisiae.* Both proteins belong to the allantoate permease family while the most sequence-similar hit (with PIDE = 31%) was the gustatory and odorant receptor 22 (Q7PMG3) from the insect *Anopheles gambiae* belonging to the gustatory receptor family. Experimental 3D structures were

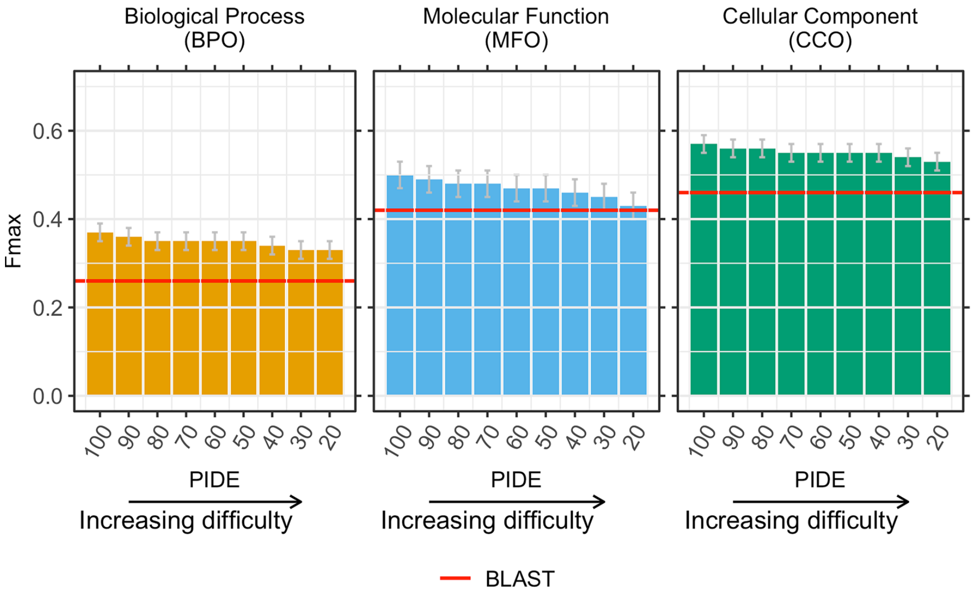

Figure 2. **Embedding-based transfer succeeded for rather diverged proteins.** After establishing the low correlation between embedding- and sequence-similarity, we tested how the level of percentage pairwise sequence identity (PIDE, x-axes) between the query (protein without known GO terms) and the transfer database (proteins with known GO terms, here subsets of *GOA2017*) affected the performance of the embedding-based transfer. Technically, we achieved this by removing proteins above a certain threshold in PIDE (decreasing toward right) from the transfer database. The y-axes showed the F_{max} score as compiled by CAFA3[19]. If embedding similarity and sequence identity correlated, our method would fall to the level of the reddish lines marked by BLAST. On the other hand, if the two were completely uncorrelated, the bars describing embedding-transfer would all have the same height (at least within the standard errors marked by the gray vertical lines at the upper end of each bar), i.e., embedding-based transfer would be completely independent of the sequence similarity between query and template (protein of known function). The observation that all bars tended to fall toward the right implied that embedding and sequence identity correlated (although for CCO, F_{max} remained within the 95% confidence interval of F_{max} for *GOA2017-100*). The observation that our method remained mostly above the baseline predictions demonstrates that embeddings capture important orthogonal information. Error bars indicate 95% confidence intervals.

not available for any of the three proteins. However, comparative modeling using Swiss-Model[36] revealed that both the target and the hit based on SeqVec were mapped to the same template (root-mean-square deviation (RMSD) = 0.3 Å) (Fig. S6a) while the hit based on sequence similarity was linked to a different structure (with RMSD = 16.8 Å) (Fig. S6b). Similarly, for the GDSL esterase/lipase At3g48460 (Q9STM6) from *Arabidopsis thaliana*, the closest hit in ProtBert embedding space was the GDSL esterase/lipase 2 (Q9SYF0) also from *Arabidopsis thaliana* while the most sequence-similar hit was the UDP-glucose 4-epimerase (Q564Q1) from *Caenorhabditis elegans*. The target and the embedding-based hit are both hydrolases belonging to the same CATH superfamily while the sequence-based hit is an isomerase and not annotated to any CATH superfamily. Comparative modeling suggested similar structures for target and embedding hit (RMSD = 2.9 Å) (Fig. S6c) while the structure found for the sequence-based hit similarity was very different (RMSD = 26 Å) (Fig. S6d). This suggested embeddings to capture structural features better than just raw sequences. Homology-based inference depends on many parameters that can especially affect the resulting sequence alignment for distantly related proteins. Possibly, embeddings are more robust in identifying those more distant evolutionary relatives.

Embedding-based transfer benefited from non-experimental annotations. Unlike the data set made available by CAFA3, annotations in our *GOA2017* data set were not limited to experimentally verified annotations. Instead, they included annotations inferred by computational biology, homology-based inference, or by "author statement evidence", i.e., through information from publications. Using *GOA2017X,* the subset of *GOA2017-100* containing only experimental terms, our method reached F_{max} = 31 ± 2% (P = 28 ± 2%, R = 34 ± 2%), 51 ± 3% (P = 53 ± 3%, R = 49 ± 3%), and 56 ± 2% (P = 55 ± 3%, R = 57 ± 3%) for BPO, MFO, and CCO, respectively. Compared to using *GOA2017-100*, the performance dropped significantly for BPO (F_{max} = 37 ± 2% for *GOA2017-100,* Table 1); it decreased slightly (within 95% confidence interval) for CCO (F_{max} = 57 ± 2% for *GOA2017-100,* Table 1); and it increased slightly (within 95% confidence interval) for MFO (F_{max} = 50 ± 3% for *GOA2017-100*, Table 1). Thus, less reliable annotations might still help, in particular for BPO. Annotations for

BPO may rely more on information available from publications that is not as easily quantifiable experimentally as annotations for MFO or CCO.

Many of the non-experimental annotations constituted sequence-based annotation transfers. Thus, non-experimental annotations might have helped because they constituted an implicit merger of sequence and embedding transfer. Adding homologous proteins might "bridge" sequence and embedding space by populating embedding space using annotations transferred from sequence space. The weak correlation between both spaces supported this speculation because protein pairs with very similar sequences may differ in their embeddings and vice versa.

Improving annotations from 2017 to 2020 increased performance significantly. For CAFA3 comparisons, we only used data available before the CAFA3 submission deadline. When running new queries, annotations will be transferred from the latest GOA. We used *GOA2020-100* (from 02/2020 removing the CAFA3 targets) to assess how the improvement of annotations from 2017 to 2020 influenced annotation transfer (Table 1). On *GOA2020-100,* SeqVec embedding-based transfer achieved $F_{max} = 50 \pm 2\%$ ($P = 50 \pm 3\%$, $R = 50 \pm 3\%$), $60 \pm 3\%$ ($P = 52 \pm 3\%$, $R = 71 \pm 3\%$), and $65 \pm 2\%$ ($P = 57 \pm 3\%$, $R = 75 \pm 3\%$) for BPO, MFO, and CCO, respectively, for the *NK* evaluation mode (Table 1). This constituted a substantial increase over *GOA2017-100* (Table 1).

The large performance boost between *GOA2017* and *GOA2020* suggested the addition of many relevant GO annotations. However, for increasingly diverged pairs (Q,T), we observed a much larger drop in F_{max} than for *GOA2017* (Fig. 2, Fig. S4). In the extreme, *GOA2020-20* (PIDE(Q,T) < 20%) with $F_{max} = 33 \pm 2\%$ (BPO), $44 \pm 2\%$ (MFO), and $54 \pm 2\%$ (CCO) fell to the same level as *GOA2017-20* (Figs. 2, S4). These results suggested that many of the relevant GO annotations were added for proteins sequence-similar to those with existing annotations. Put differently, many helpful new experiments simply refined previous computational predictions.

Running BLAST against *GOA2020-100* for sequence-based transfer (choosing the hit with the highest PIDE) showed that sequence-transfer also profited from improved annotations (difference in F_{max} values for BLAST in Table 1). However, while F_{max} scores for embedding-based transfer increased the most for BPO, those for sequence-based transfer increased most for MFO. Embedding-transfer still outperformed BLAST for the *GOA2020-100* set (Fig. S4c).

Even when constraining annotation transfer to sequence-distant pairs, our method outperformed BLAST against *GOA2020-100* in terms of F_{max} at least for BPO and for higher levels of PIDE in MFO/CCO (Fig. S4c). However, comparing the results for BLAST on the *GOA2020-100* set with the performance of our method for subsets of very diverged sequences (e.g. PIDE < 40% for *GOA2020-40*) under-estimated the differences between sequence- and embedding-based transfer, because the two approaches transferred annotations from different data sets. For a more realistic comparison, we re-ran BLAST only considering hits below certain PIDE thresholds (for comparability we could not do this for CAFA3). As expected, performance for BLAST decreased with PIDE (Fig. S4 lighter bars), e.g., for PIDE < 20%, F_{max} fell to 8% for BPO, 10% for MFO, and 11% for CCO (Fig. S4c lighter bars) largely due to low coverage, i.e., most queries had no hit to transfer annotations from. At this level (and for the same set), the embedding-based transfer proposed here, still achieved values of $33 \pm 2\%$ (BPO), $44 \pm 2\%$ (MFO), and $54 \pm 2\%$ (CCO). Thus, our method made reasonable predictions at levels of sequence identity for which homology-based inference (BLAST) failed completely.

Performance confirmed by new proteins. Our method and especially the threshold to transfer a GO term were "optimized" using the CAFA3 targets. Without any changes in the method, we tested a new data set of 298 proteins, *GOA2020-new*, with proteins for which experimental GO annotations have been added since the CAFA4 submission deadline (02/2020; Method). Using the thresholds optimized for CAFA3 targets (0.35 for BPO, 0.28 for MFO, 0.29 for CCO, Fig. 3), our method reached $F_1 = 50 \pm 11\%$, $54 \pm 5\%$, and $66 \pm 8\%$ for BPO, MFO, and CCO, respectively. For BPO and CCO, the performance was similar to that for the CAFA3 targets; for MFO it was slightly below but within the 95% CI (Table 1). For yet a different set, submitted for MFO to CAFA4, the first preliminary evaluation published during ISMB2020[37], also suggested our approach to make it to the top-ten, in line with the *post facto* CAFA3 results presented here.

Embedding similarity influenced performance. Homology-based inference works best for pairs with high PIDE. Analogously, we assumed embedding-transfer to be best for pairs with high embedding similarity, i.e., low Euclidean distance (Eq. 4). We used this to define a reliability index (RI, Eq. 5). For the *GOA2020-100* set, the minimal RI was 0.24. The CAFA evaluation determined 0.35 for BPO, 0.28 for MFO, and 0.29 for CCO as thresholds leading to optimal performance as measured by F_{max} (Fig. 3 dashed grey lines marked these thresholds). For all ontologies, precision and recall were almost constant for lower RIs (up to ~ 0.3). For higher RIs, precision increased, and recall decreased as expected (Fig. 3). While precision increased up to 82% for BPO, 91% for MFO, and 70% for CCO, it also fluctuated for high RIs (Fig. 3). This trend was probably caused by the low number of terms predicted at these RIs. For CCO, the RI essentially did not correlate with precision. This might point to a problem in assessing annotations for which the trivial Naïve method reached values of $F_{max} \sim 55\%$ outperforming most methods. Possibly, some prediction of the type "organelle" is all that is needed to achieve a high F_{max} in this ontology.

Similar performance for different embeddings. We compared embeddings derived from two different language models (LMs). So far, we used embeddings from SeqVec[20]. Recently, ProtBert, a transformer-based approach using a masked language model objective (Bert[38]) instead of auto-regression and more protein sequences (BFD[39,40]) during pre-training, was shown to improve secondary structure prediction[21]. Replacing

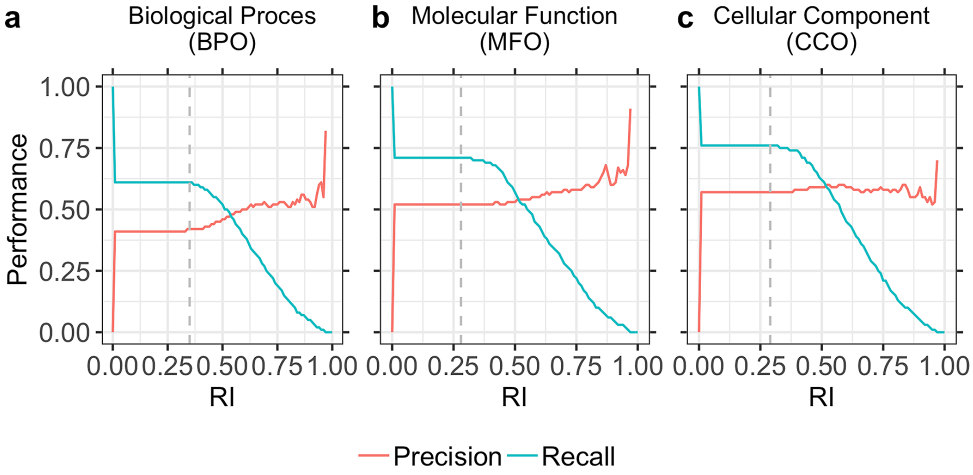

Figure 3. Precision and recall for different reliability indices (RIs). We defined a reliability index (RI) measuring the strength of a prediction (Eq. 5), i.e., for the embedding proximity. Precision (Eq. 1) and recall (Eq. 2) were almost constant for lower RIs (up to ~ 0.3) for all three ontologies (BPO, MFO, CCO). For higher RIs, precision increased while recall dropped. However, due to the low number of terms predicted at very high RIs (> 0.8), precision fluctuated and was not fully correlated with RI. Panel (**a**) shows precision and recall for BPO, panel (**b**) for MFO, and panel (**c**) for CCO. Dashed vertical lines marked the thresholds used by the CAFA3 tool to compute F_{max}: 0.35 for BPO, 0.28 for MFO, and 0.29 for CCO. At least for BPO and MFO higher RI values correlated with higher precision, i.e., users could use the RI to estimate how good the predictions are likely to be for their query, or to simply scan only those predictions more likely to be correct (e.g. RI > 0.8).

SeqVec by ProtBert embeddings to transfer annotations, our approach achieved similar F_{max} scores (Table 1). In fact, the ProtBert F_{max} scores remained within the 95% confidence intervals of those for SeqVec (Table 1). Similar results were observed when using *GOA2017-100* (Table 1).

On the one hand, the similar performance for both embeddings might indicate that both LMs extracted equally beneficial aspects of function, irrespective of the underlying architecture (LSTMs in SeqVec, transformer encoders in ProtBert) or training set (SeqVec used UniRef50 with ~ 33 M proteins, ProtBert used BFD with ~ 2.1B proteins). On the other hand, the similar F_{max} scores might also highlight that important information was lost when averaging over the entire protein to render fixed-size vectors. The similarity in F_{max} scores was less surprising given the high correlation between SeqVec and ProtBert embeddings (ρ = 0.58, p-value < 2.2e-16; Table 2). The two LMs correlated more with each other than either with PIDE (Table 2).

No gain from simple combination of embedding- and sequence-based transfer.

All three approaches toward annotation transfer (embeddings from SeqVec or ProtBert, and sequence) had strengths; although performing worse on average, for some proteins sequence-transfer performed better. In fact, analyzing the pairs for which embedding-based transfer or sequence-based transfer outperformed the other method by at least four percentage points ($|F_{max}(BLAST)-F_{max}(embeddings)| \geq 4$) illustrated the expected cases for which PIDE was high and embedding similarity low, and vice versa, along with more surprising cases for which low PIDE still yielded better predictions than relatively high embedding RIs (Fig. S5). Overall, these results (Fig. S5) again underlined that LM embeddings abstract information from sequence that are relevant for comparisons and not captured by sequences alone. However, it also indicates that even protein pairs with low embedding similarity can share similar GO terms. In fact, embedding similarity for SeqVec embeddings only weakly correlated with GO term similarity (Spearman rank coefficient ρ = 0.28, *p*-value < 2.2e−16), but proteins with identical GO annotations were on average more likely to be close than proteins with more different GO annotations (Fig. S7). The similarity of GO terms for two proteins was proxied through the Jaccard index (Eq. 7). More details are provided in the SOM.

To benefit from the cases where BLAST outperformed our approach, we tried simple combinations: firstly, we considered all terms predicted by embeddings from either SeqVec or ProtBert. Secondly, reliability scores were combined leading to higher reliability for terms predicted in both approaches than for terms only predicted by one. None of those two improved performance (Table S4, method SeqVec/ProtBert). Other simple combinations also failed so far (Table S4, method SeqVec/ProtBert/BLAST). Future work might improve performance through more advanced combinations.

Case study: embedding-based annotation transfer for three proteomes.

Due to its simplicity and speed, embedding-based annotation transfer can easily be applied to novel proteins to shed light on their potential functionality. We applied our method to the proteomes of three different proteomes: human (20,370 proteins from Swiss-Prot) as a well-researched proteome, the fungus *Armillaria ostoyae* (22,192 proteins, 0.01%

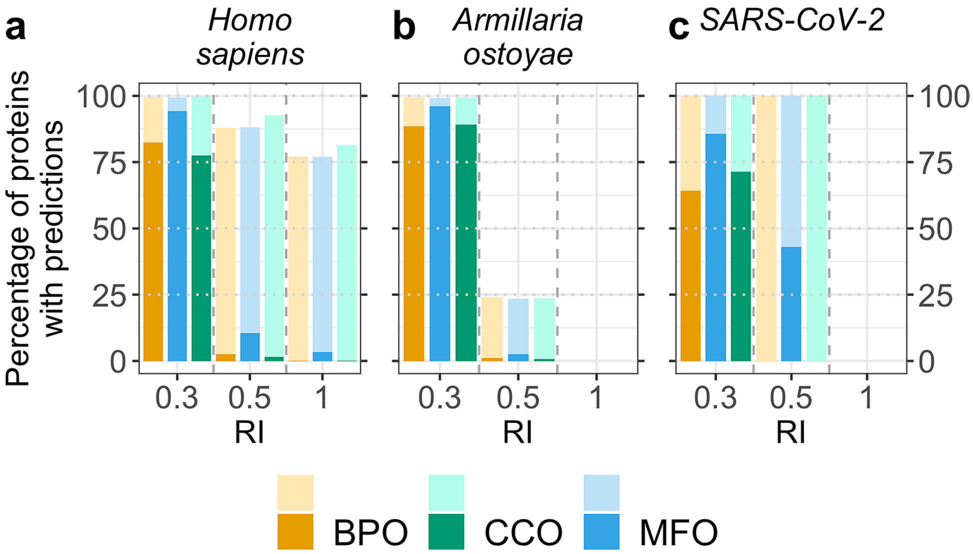

Figure 4. **Fraction of proteomes with predicted GO terms.** We applied our method to three proteomes (animal: *Homo sapiens*, fungus: *Armillaria ostoyae*, and virus: SARS-Cov-2) and monitored the fraction of proteins in each proteome for which our method predicted GO terms for different thresholds in embedding similarity (RI, Eq. 5). We show predictions for RI = 1.0 ("self-hits"), RI = 0.5 (with an expected precision/recall = 0.5), and RI = 0.3 (CAFA3 thresholds). Darker colored bars indicate predictions using *GOA2020X* as lookup set (only experimentally verified GO annotations) and lighter colors indicate predictions using *GOA2020* as lookup set (using all annotations in GOA). (**a**) The human proteome is well-studied (all 20,370 proteins are in Swiss-Prot) and for most proteins, GO annotations are available, but those annotations are largely not experimentally verified (very small, dark-colored bars vs large, lighter-colored bars at RI = 1.0). (**b**) The proteome of the fungus *Armillaria ostoyae* appears more exceptional (0.01% of the 22,192 proteins were in Swiss-Prot); at RI ≥ 0.5, predictions could be made only for 25% of the proteins when also using unverified annotations and none of the proteins already had any GO annotations. (**c**) While annotations were unknown for most proteins of the novel virus SARS-CoV-2 (no coverage at RI = 1), many annotations could be transferred from the human SARS coronavirus (SARS-CoV) and the bat coronavirus HKU3 (BtCoV) allowing GO term predictions for all proteins at reliability values ≥ 0.5.

of these in Swiss-Prot), as one of the oldest (2500 years) and largest ($4*10^5$ kg/spanning over 10 km^2) living organisms known today[41], and SARS-CoV-2, the virus causing COVID-19 (14 proteins). At RI = 1.0, annotations were inferred from proteins of this organism ("self-hits"). Using only experimentally verified annotations (lookup data set *GOA2020X*), revealed both how few proteins were directly annotated (self-hits) in these organisms and how much of the sequence-annotation gap is gapped through embedding-based inference (Fig. 4: bars with darker orange, blue, green for BPO, CCO, and MFO respectively). In particular, for self-hits, i.e., proteins with 100% pairwise sequence identity (PIDE) to the protein with known annotation, it became obvious how few proteins in human have explicit experimental annotation (sum over all around 270), while through embedding-based inference up to 80% of all human proteins could be annotated through proteins from other organisms (light bars in Fig. 4 give results for the entire *GOA2020* which is dominated by annotations not directly verified by experiment). For the other two proteomes from the fungus (*Armillaria ostoyae*) and the coronavirus (SARS-CoV-2), there were no inferences at this high level. On the other end of including all inferences as assessed through the data presented in all other figures and tables (i.e., at the default thresholds), for all three proteins most proteins could be annotated directly from experimentally verified annotations through embeddings (three left-most bars in Fig. 4 for BPO, CCO, and MFO). In fact, when including all GO annotations from GOA (lookup set *GOA2020*), almost all proteins in all three proteomes could be annotated (Fig. 4: lighter colored left-most bars close to fraction of 1, i.e., all proteins). For SARS-CoV-2, our method reached 100% coverage (prediction for all proteins) already at RI≥0.5 (Fig. 4c, lighter colors, middle bars) through well-studied, similar viruses such as the human SARS coronavirus (SARS-CoV). RI = 0.5 represent roughly a precision and recall of 50% for all three ontologies (Fig. 3). For *Armillaria ostoyae*, almost no protein was annotated through self-hits even when using unverified annotations (Fig. 4b: no bar at RI = 1). At RI = 0.5, about 25% of the proteins were annotated.

Case study: embedding-based annotation transfer for SARS-CoV-2 proteome. Given the relevance of SARS-CoV-2, we did not only apply our method to predict GO terms (BPO, MFO, and CCO) for all 14 SARS-CoV-2 proteins (taken from UniProt[3]; all raw predictions were made available as additional files named predictions_$emb_$ont.txt replacing the variables $emb and $ont as follows: $emb = seqvec|protbert, and \$ont = bpo|mfo|cco), but also investigated the resulting annotations further. While the two replicase polyproteins pp1a and pp1ab can also be split further into up to 12 non-structural proteins resulting in 28 proteins[42], we used the definition from UniProt identifying 14 different proteins.

Step 1: confirmation of known annotations. Out of the 42 predictions (14 proteins in 3 ontologies), 12 were based on annotation transfers using proteins from the human SARS coronavirus (SARS-CoV), and 13 on proteins from the bat coronavirus HKU3 (BtCoV). CCO predictions appeared reasonable with predicted locations mainly associated with the virus (e.g., viral envelope, virion membrane) or the host (e.g., host cell Golgi apparatus, host cell membrane). Similarly, MFO predictions often matched well-known annotations, e.g., the replicase polyproteins 1a and 1ab were predicted to be involved in RNA-binding as confirmed by UniProt. In fact, annotations in BPO were known for 7 proteins (in total 40 GO terms), in MFO for 6 proteins (30 GO terms), and in CCO for 12 proteins (68 GO terms). Only three of these annotations were experimentally verified. With our method, we predicted 25 out of the 40 GO terms for BPO (63%), 14/30 for MFO (47%), and 59/68 for CCO (87%). Even more annotations were similar to the known GO annotations but were more or less specific (Table S5 summarized all predicted and annotated GO leaf terms, the corresponding names can be found in the additional files predictions_\$emb_\$ont.txt).

Step 2: new predictions. Since the GO term predictions matched well-characterized proteins, predictions might provide insights into the function of proteins without or with fewer annotations. For example, function and structure of the non-structural protein 7b (Uniprot identifier P0DTD8) are not known except for a transmembrane region of which the existence was supported by the predicted CCO annotation "*integral component of the membrane*" and "*host cell membrane*". This CCO annotation was also correctly predicted by the embedding-based transfer from an *Ashbya gossypii* protein. Additionally, we predicted "*transport of virus in host*, *cell to cell*" for BPO and "*proton transmembrane transporter activity*" for MFO. This suggested non-structural protein 7b to play a role in transporting the virion through the membrane into the host cell. Visualizing the leaf term predictions in the GO hierarchy could help to better understand very specific annotations. For the BPO annotation of the non-structural protein 7b, the tree revealed that this functionality constituted two major aspects: The interaction with the host and the actual transport to the host (Fig. S10). To visualize the predicted terms in the GO hierarchy, for example the tool NaviGO[43] can be used which can help to interpret the GO predictions given for the SARS-CoV-2 proteins here.

Comparing annotation transfers based on embeddings from SeqVec and from ProtBert showed that 16 of the 42 predictions agreed for the two different language models (LMs). For five predictions, one of the two LMs yielded more specific annotations, e.g., for the nucleoprotein (Uniprot identifier P0DTC9) which is involved in viral genome packaging and regulation of viral transcription and replication. For this protein, SeqVec embeddings found no meaningful result, while ProtBert embeddings predicted terms such as "*RNA polymerase II preinitiation complex*" and "*positive regulation of transcription by RNA polymerase II*" fitting to the known function of the nucleoprotein. This example demonstrated how the combination of results from predictions using different LMs may refine GO term predictions.

Conclusions

We introduce a new concept for the prediction of GO terms, namely the annotation transfer based on similarity of embeddings obtained from deep learning language models (LMs). This approach conceptually replaces sequence information by complex embeddings that capture some non-local information beyond sequence similarity. The underlying LMs (SeqVec & ProtBert) are highly involved and complex, and their training is time-consuming and data intensive. Once that is done, those pre-trained LMs can be applied, their abstracted understanding of the language of life as captured by protein sequences can be transferred to yield an extremely simple, yet effective novel method for annotation transfer. This novel prediction method complements homology-based inference. Despite its simplicity, this new method outperformed by several margins of statistically significance homology-based inference ("BLAST") with F_{max} values of BPO + **11** ± 2% (F_{max}(embedding)-F_{max}(sequence)), MFO + **8** ± 3%, and CCO + **11** ± 2% (Table 1, Fig. 1); it even might have reached the top ten, had it participated at CAFA3 (Fig. 1). Embedding-based transfer remained above the average for sequence-based transfer even for protein pairs with PIDE < 20% (Fig. 2), i.e., embedding similarity worked for proteins that diverged beyond the recognition in pairwise alignments (Figs. S2 & S3). Embedding-based transfer is also blazingly fast to compute, i.e., around 0.05 s per protein. The only time-consuming step is computing embeddings for all proteins in the lookup database which needs to be done only once; it took about 30 min for the entire human proteome. GO annotations added from 2017 to 2020 improved both sequence- and embedding-based annotation transfer significantly (Table 1). Another aspect of the simplicity is that, at least in the context of the CAFA3 evaluation, the choice of none of the two free parameters really mattered: embeddings from both LMs tested performed, on average, equally, and the number of best hits (k-nearest neighbors) did not matter much (Table S2). The power of this new concept is generated by the degree to which embeddings implicitly capture important information relevant for protein structure and function prediction. One reason for the success of our new concept was the limited correlation between embeddings and sequence (Table 2). Additionally, the abstraction of sequence information in embeddings appeared to make crucially meaningful information readily available (Fig. S6). This implies that embeddings have the potential to revolutionize the way sequence comparisons are carried out.

Methods

Generating embedding space.

The embedding-based annotation transfer introduced here requires each protein to be represented by a fixed-length vector, i.e., a vector with the same dimension for a protein of 30 and another of 40,000 residues (maximal sequence length for ProtBert). To this end, we used SeqVec[20] to represent each protein in our data set by a fixed size embedding. SeqVec is based on ELMo[44] using a stack of LSTMs[45] for auto-regressive pre-training[46,47] i.e., predicting the next token (originally a word in a sentence, here an amino acid in a protein sequence), given all previous tokens. Two independent stacks of LSTMs process the sequence from both directions. During pre-training, the two directions are joined by summing their losses; concatenating the hidden states of both directions during inference lets supervised tasks capture bi-directional context. For SeqVec, three layers, i.e., one uncontextualized CharCNN[48] and two bi-directional LSTMs, were trained on each protein in UniRef50 (UniProt[3] clustered at 50% PIDE resulting in ~ 33 M proteins). In order to increase regularization, the weights of the token representation (CharCNN) as well as the final Softmax layer were shared between the two LSTM directions, and a 10% dropout rate was applied. For SeqVec, the CharCNN as well as each LSTM has a hidden state of size 512, resulting in a total of 93 M free parameters. As only unlabeled data (no phenotypical data) was used (self-supervised training), the embeddings could not capture any explicit information such as GO numbers. Thus, SeqVec does not need to be retrained for subsequent prediction tasks using the embeddings as input. The hidden states of the pre-trained model are used to extract features. Corresponding to its hidden state size, SeqVec outputs for each layer and each direction a 512-dimensional vector; in this work, only the forward and backward passes of the first LSTM layer were extracted and concatenated into a matrix of size L * 1024 for a protein with L residues. While the auto-regressive pre-training only allowed to gather contextual information from either direction, the concatenation of the representations allowed our approach to benefit from bi-directional context. A fixed-size representation was then derived by averaging over the length dimension, resulting in a vector of size 1024 for each protein (Fig. S11). This simple way of information pooling (also called *global average pooling*) outperformed in many cases more sophisticated methods in NLP[49] and showed competitive performance in bioinformatics for some tasks[20,21,26]. Based on experience from NLP[49,50], we also investigated the effect of using a different pooling strategy, i.e., maximum pooling, to derive fixed size representations from SeqVec embeddings.

To evaluate the effect of using different LMs to generate the embeddings, we also used a transformer-based LM trained on protein sequences (ProtBert-BFD[21], here simply referred to as *ProtBert*). ProtBert is based on the LM BERT[38] (Bidirectional Encoder Representations from Transformers[51]) which processes sequential data through the self-attention mechanism[52]. Self-attention compares all tokens in a sequence to all others in parallel, thereby capturing long-range dependencies better than LSTMs. BERT also replaced ELMo's auto-regressive objective by masked language modeling during pre-training, i.e., reconstructing corrupted tokens from the input, which enables to capture bi-directional context. ProtBert was trained with 30 attention layers, each having 16 attention heads with a hidden state size of 1024 resulting in a total of 420 M free parameters which were optimized on 2.1B protein sequences (BFD)[39,40] which is 70 times larger than UniRef50. The output of the last attention layer of ProtBert was used to derive a 1024-dimensional embedding for each residue. As for SeqVec, the resulting L * 1024 matrix was pooled by averaging over protein length providing a fixed-size vector of dimension 1024 for each protein. Usually, BERT's special CLS-token is used for sequence-classification tasks[38] as it is already optimized during pre-training on summarizing sequence information by predicting whether two sentences are consecutive in a document or not. In the absence of such a concept for proteins, this second loss was dropped from ProtBert's pre-training rendering the CLS token without further fine-tuning on supervised tasks uninformative.

Embeddings derived from LMs change upon retraining the model with a different random seed, even using the same data and hyper-parameters. They are likely to change more substantially when switching the training data or tuning hyper-parameters. As retraining LMs is computationally (and environmentally) expensive, we leave assessing the impact of fine-tuning LMs to the future.

Generating the embeddings for all human proteins using both SeqVec and ProtBert allowed estimating the time required for the generation of the input to our new method. Using a single Nvidia GeForce GTX1080 with 8 GB vRAM and dynamic batching (depending on the sequence length), this took, on average, about 0.05 s per protein[21].

Data set. To create a database for annotation transfer, we extracted protein sequences with annotated GO terms from the Gene Ontology Annotation (GOA) database[53–55] (containing 29,904,266 proteins from UniProtKB[3] in February 2020). In order to focus on proteins known to exist, we only extracted records from Swiss-Prot[56]. Proteins annotated only at the ontology roots, i.e. proteins limited to "GO:0003674" (molecular_function), "GO:0008150" (biological_process), or "GO:0005575" (cellular_component) were considered meaningless and were excluded. The final data set *GOA2020* contained 295,558 proteins (with unique identifiers, IDs) described by 31,485 different GO terms. The GO term annotation for each protein includes all annotated terms and all their parent terms. Thereby, proteins are, on average, annotated by 37 terms in BPO, 14 in MFO, and 9 in CCO. Counting only leaves brought the averages to 3 in BPO, 2 in MFO, and 3 in CCO.

For comparison to methods that contributed to CAFA3[19], we added another data set *GOA2017* using the GOA version available at the submission deadline of CAFA3 (Jan 17, 2017). After processing (as for *GOA2020*), *GOA2017* contained 307,287 proteins (unique IDs) described by 30,124 different GO terms. While we could not find a definite explanation for having fewer proteins in the newer database (*GOA2020* 295 K proteins vs. *GOA2017* with 307 K), we assume that it originated from major changes in GO including the removal of obsolete and inaccurate annotations and the refactoring of MFO[2].

The above filters neither excluded GOA annotations inferred from phylogenetic evidence and author statements nor those based on computational analysis. We constructed an additional data set, *GOA2017X* exclusively containing proteins annotated in Swiss-Prot as experimental (evidence codes EXP, IDA, IPI, IMP, IGI, IEP, TAS, or IC) following the CAFA3 definition[19]. We further excluded all entries with PIDE = 100% to any CAFA3 target bringing *GOA2017X* to 303,984 proteins with 28,677 different GO terms.

Performance evaluation. The targets from the CAFA3 challenge[19] were used to evaluate the performance of our new method. Of the 130,827 targets originally released for CAFA3, experimental GO annotations were obtained for 3328 proteins at the point of the final benchmark collection in November 2017[19]. This set consisted of the following subsets with experimental annotations in each sub-hierarchy of GO: BPO 2145, MFO 1101, and CCO 1097 (more details about the data set are given in the original CAFA3 publication[19]).

We used an additional data set, dubbed *GOA2020-new*, containing proteins added to GOA after February 2020, i.e., the point of accession for the GOA set used during the development of our method in preparation for CAFA4. This set consisted of 298 proteins with experimentally verified GO annotations and without any identical hits (i.e. 100% PIDE) in the lookup set *GOA2020*.

In order to expand the comparison of the transfer based on sequence- and embedding similarity, we also reduced the redundancy through applying CD-HIT and PSI-CD-HIT[57] to the *GOA2020* and *GOA2017* sets against the evaluation set at thresholds θ of PIDE = 100, 90, 80, 70, 60, 50, 40, 30 and 20% (Table S6 in the Supporting Online Material (SOM) for more details about these nine subsets).

We evaluated our method against two baseline methods used at CAFA3, namely *Naïve* and *BLAST*, as well as, against CAFA3′s top ten[19]. We computed standard performance measures. True positives (TP) were GO terms predicted above a certain reliability (RI) threshold (Method below), false positives (FP) were GO terms predicted but not annotated, and false negatives (FN) were GO terms annotated but not predicted. Based on these three numbers, we calculated precision (Eq. 1), recall (Eq. 2), and F1 score (Eq. 3) as follows.

$$P = precision = \frac{TP}{TP + FP} \tag{1}$$

$$R = recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = 2 \bullet \frac{Precision \bullet Recall}{Precision + Recall} \tag{3}$$

The F_{max} value denoted the maximum F1 score achievable for any threshold in reliability (RI, Eq. 5). This implies that the assessment fixes the optimal value rather than the method providing this value. Although this arguably over-estimates performance, it has evolved to a quasi-standard of CAFA; the publicly available CAFA Assessment Tool[18,19] calculated F_{max} for the CAFA3 targets in the same manner as for the official CAFA3 evaluation. If not stated otherwise, we reported precision and recall values for the threshold leading to F_{max}.

CAFA3 assessed performance separately for two sets of proteins for all three ontologies: (i) proteins for which no experimental annotations were known beforehand (no-knowledge, NK evaluation mode) and (ii) proteins with some experimental annotations in one or two of the other ontologies (limited-knowledge, LK evaluation mode)[9,19]. We also considered these sets separately in our assessment. CAFA3 further distinguished between *full* and *partial evaluation* with *full evaluation* penalizing if no prediction was made for a certain protein, and *partial evaluation* restricting the assessment to the subset of proteins with predictions[19]. Our method predicted for every protein; thus, we considered only the *full evaluation*. Also following CAFA3, symmetric 95% confidence intervals were calculated as error estimates assuming a normal distribution and 10,000 bootstrap samples estimated mean and standard deviation.

Method: annotation transfer through embedding similarity. For a given query protein Q, GO terms were transferred from proteins with known GO terms (sets *GOA2020* and *GOA2017*) through an approach similar to the k-nearest neighbor algorithm (k-NN)[58]. For the query Q and for all proteins in, e.g., *GOA2020*, the SeqVec[20] embeddings were computed. Based on the Euclidean distance between two embeddings *n* and *m* (Eq. 4), we extracted the *k* closest hits to the query from the database where *k* constituted a free parameter to optimize.

$$d(n, m) = \sqrt{\sum_{i=1}^{1024} (n_i - m_i)^2} \tag{4}$$

In contrast to standard k-NN algorithms, all annotations from all hits were transferred to the query instead of only the most frequent one[58]. When multiple pairs reached the same distance, all were considered, i.e., for a given *k*, more than *k* proteins might be considered for the GO term prediction. The calculation of the pairwise Euclidean distances between queries and all database proteins and the subsequent nearest neighbor extraction was accomplished very efficiently. For instance, the nearest-neighbor search of 1000 query proteins against GOA20* with about 300,000 proteins took on average only about 0.005 s per query on a single i7-6700 CPU, i.e., less than two minutes for all human proteins.

Converting the Euclidean distance enabled to introduce a reliability index (RI) ranging from 0 (weak prediction) to 1 (confident prediction) for each predicted GO term *p* as follows:

$$RI(p) = \frac{1}{k} \sum_{i=1}^{l} \frac{0.5}{0.5 + d(q, n_i)} \tag{5}$$

with k as the overall number of hits/neighbors, l as the number of hits annotated with the GO term p and the distance $d(q, n_i)$ between query and hit being calculated according to Eq. (4).

Proteins represented by an embedding identical to the query protein (d = 0) led to RI = 1. Since the RI also takes into account, how many proteins l in a list of k hits are annotated with a certain term p (Eq. 5), predicted terms annotated to more proteins (larger l) have a higher RI than terms annotated to fewer proteins (smaller l). As this approach accounts for the agreement of the annotations between the k hits, it requires the RI to be normalized by the number of considered neighbors k, making it not directly comparable for predictions based on different values for k. On top, if different embeddings are used to identify close proteins, RI values are not directly comparable, because embeddings might be on different scales.

Instead of assessing embedding proximity through the Euclidean distance, the embedding field typically uses the cosine distance (Eq. 6):

$$d_{cosine}(n, m) = 1 - \frac{\sum_{i=1}^{1024} n_i m_i}{\sqrt{\sum_{i=1}^{1024} n_i^2} \bullet \sqrt{\sum_{i=1}^{1024} m_i^2}} \tag{6}$$

Our initial assessment suggested cosine and Euclidean distance to perform alike, and we chose to use the metric more familiar to structural biologists, namely the Euclidean distance throughout this analysis.

GO term similarity. We measured the similarity between two sets of GO annotations A and B through the Jaccard index (Eq. 7) where $|A \cap B|$ is the number of GO terms present in both sets and $|A \cup B|$ is the number of GO terms present in at least one of the sets (duplicates are only counted once):

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{7}$$

Correlation analysis. We analyzed the correlation between sequence identity and embedding similarity through the Spearman's rank correlation coefficient because our data was neither distributed normally, nor were the two measures for similarity measures linear. In contrast to, e.g. Pearson correlation, Spearman does not assume a normal distribution and detects monotonic instead of linear relations[59,60].

Availability. GO term predictions using embedding similarity for a certain protein sequence can be performed through our publicly available webserver: https://embed.protein.properties/. The source code along with all embeddings for *GOA2020* and *GOA2017*, and the CAFA3 targets are also available on GitHub: https://github.com/Rostlab/goPredSim (more details in the repository). In addition to reproducing the results, the source code also allows calculating embedding similarity using cosine distance.

Data availability

The source code and the embedding sets for target proteins and lookup databases are publicly available as a GitHub repository. GO term predictions for the SARS-CoV-2 proteins are provided as additional files and in the GitHub repository.

Received: 7 September 2020; Accepted: 24 December 2020
Published online: 13 January 2021

References

1. Krebs, H. A. & Johnson, W. A. Metabolism of ketonic acids in animal tissues. *Biochem J* **31**, 645–660. https://doi.org/10.1042/bj0310645 (1937).
2. The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Res* **47**, D330–D338. https://doi.org/10.1093/nar/gky1055 (2019).
3. UniProt Consortium. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Res.* **47**, D506–D515. https://doi.org/10.1093/nar/gky1049 (2019).
4. Hirst, J. D. & Sternberg, M. J. E. Prediction of structural and functional features of protein and nucleic acid sequences by artificial neural networks. *Biochemistry* **31**, 615–623 (1992).
5. Rost, B., Liu, J., Nair, R., Wrzeszczynski, K. O. & Ofran, Y. Automatic prediction of protein function. *Cell. Mol. Life Sci.* **60**, 2637–2650 (2003).
6. Leslie, C., Eskin, E., Weston, J. & Noble, W. S. Mismatch string kernels for SVM protein classification. *Bioinformatics*, in press (2003).
7. Ofran, Y., Punta, M., Schneider, R. & Rost, B. Beyond annotation transfer by homology: novel protein-function prediction methods to assist drug discovery. *Drug Discov. Today* **10**, 1475–1482 (2005).
8. Hamp, T. *et al.* Homology-based inference sets the bar high for protein function prediction. *BMC Bioinform.* **14**(Suppl 3), S 7. https://doi.org/10.1186/1471-2105-14-S3-S7 (2013).
9. Radivojac, P. *et al.* A large-scale evaluation of computational protein function prediction. *Nat. Methods* **10**, 221–227. https://doi.org/10.1038/nmeth.2340 (2013).
10. Cozzetto, D., Minneci, F., Currant, H. & Jones, D. T. FFPred 3: feature-based function prediction for all Gene Ontology domains. *Sci. Rep.* **6**, 31865. https://doi.org/10.1038/srep31865 (2016).
11. Kulmanov, M. & Hoehndorf, R. DeepGOPlus: improved protein function prediction from sequence. *Bioinformatics* **36**, 422–429. https://doi.org/10.1093/bioinformatics/btz595 (2020).
12. Zuckerkandl, E. Evolutionary processes and evolutionary noise at the molecular level. *J. Mol. Evol.* **7**, 269–311 (1976).
13. Nakai, K. & Horton, P. PSORT: a program for detecting sorting signals in proteins and predicting their subcellular localization. *Trends Biochem. Sci.* **24**, 34–36 (1999).

14. Nair, R. & Rost, B. Sequence conserved for sub-cellular localization. *Protein Sci.* **11**, 2836–2847 (2002).
15. Goldberg, T. *et al.* LocTree3 prediction of localization. *Nucleic Acids Res.* **42**, W350-355. https://doi.org/10.1093/nar/gku396 (2014).
16. Qiu, J. *et al.* ProNA2020 predicts protein-DNA, protein-RNA, and protein-protein binding proteins and residues from sequence. *J. Mol. Biol.* **432**, 2428–2443. https://doi.org/10.1016/j.jmb.2020.02.026 (2020).
17. Goldberg, T., Rost, B. & Bromberg, Y. Computational prediction shines light on type III secretion origins. *Sci. Rep.* **6**, 34516. https://doi.org/10.1038/srep34516 (2016).
18. Jiang, Y. *et al.* An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biol.* **17**, 184. https://doi.org/10.1186/s13059-016-1037-6 (2016).
19. Zhou, N. *et al.* The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biol.* **20**, 244. https://doi.org/10.1186/s13059-019-1835-8 (2019).
20. Heinzinger, M. *et al.* Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinform.* **20**, 723. https://doi.org/10.1186/s12859-019-3220-8 (2019).
21. Elnaggar, A. *et al.* ProtTrans: towards cracking the language of life's code through self-supervised deep learning and high performance computing. *bioRxiv* (2020).
22. Mikolov, T., Cheng, K., Corrado, G. & Dean, J. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, 2013, Workshop Track Proceedings.*
23. Allen, C. & Hospedales, T. Analogies Explained: Towards Understanding Word Embeddings. In *Proceedings of the 36th International Conference on Machine Learning.* 223–231 (PMLR).
24. Brokos, G.-I., Malakasiotis, P. & Androutsopoulos, I. Using Centroids of Word Embeddings and Word Mover's Distance for Biomedical Document Retrieval in Question Answering. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing.* 114–118 (Association for Computational Linguistics).
25. Kusner, M. J., Sun, Y., Kolkin, N. I. & Weinberger, K. Q. From Word Embeddings to Document Distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37.*
26. Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, 622803. https://doi.org/10.1101/622803 (2020).
27. Vig, J. *et al.* BERTology meets Biology: Interpreting Attention in Protein Language Models. *arXiv* (2020).
28. R Core Team (R Foundation for Statistical Computing, 2017).
29. Rost, B. Twilight zone of protein sequence alignments. *Protein Eng.* **12**, 85–94 (1999).
30. Rost, B. Enzyme function less conserved than anticipated. *J. Mol. Biol.* **318**, 595–608 (2002).
31. Mika, S. & Rost, B. Protein–protein interactions more conserved within species than across species. *PLoS Comput. Biol.* **2**, e79. https://doi.org/10.1371/journal.pcbi.0020079 (2006).
32. Clark, W. T. & Radivojac, P. Analysis of protein function and its prediction from amino acid sequence. *Proteins* **79**, 2086–2096. https://doi.org/10.1002/prot.23029 (2011).
33. Rost, B. Protein structures sustain evolutionary drift. *Fold Des.* **2**, S19–S24 (1997).
34. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410. https://doi.org/10.1016/S0022-2836(05)80360-2 (1990).
35. Steinegger, M. & Söding, J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.* **35**, 1026–1028. https://doi.org/10.1038/nbt.3988 (2017).
36. Waterhouse, A. *et al.* SWISS-MODEL: homology modelling of protein structures and complexes. *Nucleic Acids Res.* **46**, W296–W303. https://doi.org/10.1093/nar/gky427 (2018).
37. El-Mabrouk, N. & Slonim, D. K. ISMB 2020 proceedings. *Bioinformatics* **36**, i1–i2. https://doi.org/10.1093/bioinformatics/btaa537 (2020).
38. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* 4171–4186 (Association for Computational Linguistics).
39. Steinegger, M., Mirdita, M. & Söding, J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods* **16**, 603–606. https://doi.org/10.1038/s41592-019-0437-4 (2019).
40. Steinegger, M. & Söding, J. Clustering huge protein sequence sets in linear time. *Nat. Commun.* **9**, 2542. https://doi.org/10.1038/s41467-018-04964-5 (2018).
41. Anderson, J. B. *et al.* Clonal evolution and genome stability in a 2500-year-old fungal individual. *Proc. Biol. Sci.* **285**, 20182233. https://doi.org/10.1098/rspb.2018.2233 (2018).
42. O'Donoghue, S. I. *et al.* SARS-CoV-2 structural coverage map reveals state changes that disrupt host immunity. *bioRxiv* (2020).
43. Wei, Q., Khan, I. K., Ding, Z., Yerneni, S. & Kihara, D. NaviGO: interactive tool for visualization and functional similarity and coherence analysis with gene ontology. *BMC Bioinform.* **18**, 177. https://doi.org/10.1186/s12859-017-1600-5 (2017).
44. Peters, M. E. *et al.* Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* 2227–2237 (Association for Computational Linguistics).
45. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
46. Mousa, A. & Schuller, B. Contextual Bidirectional Long Short-Term Memory Recurrent Neural Network Language Models: a generative approach to sentiment analysis. In *Proceedings of the 15th Conference of the {E}uropean Chapter of the Association for Computational Linguistics: Volume 1, Long Papers.* 1023–1032 (Association for Computational Linguistics).
47. Peters, M., Ammar, W., Bhagavatula, C. & Power, R. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 1756–1765 (Association for Computational Linguistics).
48. Kim, Y., Jernite, Y., Sontag, D. & Rush, A. M. Character-aware Neural Language Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence.* (AAAI Press).
49. Shen, D. *et al.* Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 440–450 (Association for Computational Linguistics).
50. Conneau, A., Douwe, K., Schwenk, H., Barrault, L. & Bordes, A. Supervised Learning of Universal Sentence Representations From Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.* 670–680 (Association for Computational Linguistics).
51. Vaswani, A. *et al.* Attention is All You Need. In *Neural information processing systems conference.* (eds I Guyon *et al.*) 5998–6008 (Curran Associates, Inc.).
52. Bahdanau, D., Cho, K. H. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate in *arXiv.*
53. Camon, E. *et al.* The Gene Ontology Annotation (GOA) database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Res.* **32**, D262–266. https://doi.org/10.1093/nar/gkh021 (2004).
54. Huntley, R. P. *et al.* The GOA database: gene Ontology annotation updates for 2015. *Nucleic Acids Res.* **43**, D1057–1063. https://doi.org/10.1093/nar/gku1113 (2015).
55. *GOA*, http://www.ebi.ac.uk/GOA (2020).

56. Boutet, E., Lieberherr, D., Tognolli, M., Schneider, M. & Bairoch, A. UniProtKB/Swiss-Prot. *Methods Mol. Biol.* **406**, 89–112. https://doi.org/10.1007/978-1-59745-535-0_4 (2007).
57. Fu, L., Niu, B., Zhu, Z., Wu, S. & Li, W. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics* **28**, 3150–3152. https://doi.org/10.1093/bioinformatics/bts565 (2012).
58. Cover, T. & Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **13**, 21–27 (1967).
59. Dodge, Y. *The Concise Encyclopedia of Statistics 502–505* (Springer, New York, 2008).
60. Spearman, C. The Proof and Measurement of Association Between Two Things. *Am. J. Psychol.* **15**, 72–101 (1904).

Acknowledgements

Thanks to Tim Karl and Inga Weise (both TUM) for invaluable help with technical and administrative aspects of this work. Thanks to the organizers and participants of the CAFA challenge for their crucial contributions, in particular to Predrag Radivojac (Northeastern Boston), Iddo Friedberg (Iowa State Ames), Sean Mooney (U of Washington Seattle), Casey Green (U Penn Philadelphia, USA), Mark Wass (Kent, England), and Kimberly Reynolds (U Texas Dallas). Particular thanks to the maintainers of the Gene Ontology (in particular to Michael Ashburner—Cambridge and Suzanna Lewis – Berkeley), as well as, the Gene Ontology Annotation database (maintained by the team of Claire O'Donovan at the EBI) for their standardized vocabulary and curated data sets. Last, but not least, thanks to all maintainers of public databases and to all experimentalists who enabled this analysis by making their data publicly available. This work was supported by the Bavarian Ministry of Education through funding to the TUM, by a grant from the Alexander von Humboldt foundation through the German Ministry for Research and Education (BMBF: Bundesministerium für Bildung und Forschung), by a grant from BMBF (Software Campus, 01IS17049) as well as by a grant from Deutsche Forschungsgemeinschaft (DFG–GZ: RO1320/4–1). We gratefully acknowledge the support of NVIDIA Corporation with the donation of one Titan GPU used for this research.

Author contributions

M.L. implemented the method goPredSim and performed evaluations. M.H. provided SeqVec and ProtBert embeddings and contributed various ideas and comments to improve goPredSim and its assessment. M.L. and M.H. performed the major part of manuscript writing and figure generation. C.D. implemented the webserver allowing GO term predictions using our method through a web interface. T.O. had the initial idea to calculate correlation between sequence and embedding similarity and computed the combination of sequence- and embedding-based transfer. B.R. supervised and guided the work over the entire time and proofread the manuscript. All authors read and approved the final manuscript.

Funding

Open Access funding enabled and organized by Projekt DEAL.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-020-80786-0.

Correspondence and requests for materials should be addressed to M.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

4. Variant Effect Score Prediction without Alignments - VESPA

4.1. Preface

The emergence of SARS-CoV-2 variants stressed the demand for tools allowing to interpret the effect of single amino acid variants (SAVs) on protein function in a quick, yet reliable way. While Deep Mutational Scanning (DMS) sets continue to expand our understanding of the mutational landscape of single proteins by mutating each residue in a protein to all other amino acids, the results continue to challenge analyses due to a lack of standardization [111]. The latter is partly explained by a lack of standardization in data reporting due to the the complexity arising from the broad spectrum of protein function(s) as well as the significant impact of slight changes in the experimental setup on it.

Protein Language Models (pLMs) use the latest deep learning (DL) algorithms to leverage growing databases of protein sequences (Chapter 2 - SeqVec and [28, 96]). These methods learn directly from such large but unlabeled data by predicting missing or masked amino acids from the context of entire sequence regions. Here, we used pLM representations (embeddings) from on of our pLM, ProtT5 [28], to predict sequence conservation and SAV effects without MSAs. ProtT5 embeddings alone predicted residue conservation almost as accurately from single sequences as ConSeq [42] using MSAs (two-state MCC for ProtT5 embeddings of 0.60±0.01 vs. 0.61±0.01 for ConSeq). Inputting the conservation prediction along with BLOSUM62 substitution scores [50] and pLM mask reconstruction probabilities into a simplistic logistic regression ensemble for *Variant Effect Score Prediction without Alignments* (VESPA) predicted SAV effect magnitude without any optimization on DMS data. Comparing predictions for a standard set of 39 DMS experiments to other methods (incl. ESM-1v [112], DeepSequence [113], and GEMME [114]) revealed our approach as competitive with the state-of-the-art methods using MSA input. No method outperformed all others, neither consistently nor statistically significantly, independently of the performance measure applied (Spearman and Pearson correlation). Finally, we investigated binary effect predictions on DMS experiments for four human proteins. Overall, embedding-based methods have become competitive with methods relying on MSAs for SAV effect prediction at a fraction of the costs in computing/energy. Our method predicted SAV effects for the entire human proteome (20 k proteins) within 40 minutes on a single Nvidia Quadro RTX 8000. All methods and data sets are freely available for execution through the *bio_embeddings* package [107] and https://github.com/Rostlab/VESPA. We provide pre-computed predictions for the human proteome under https://zenodo.org/record/5905863 .

Author contribution: I contributed to the original conceptualisation and trained and evaluated the conservation prediction. Céline Marquet implemented the variant effect scoring method and performed the evaluation thereof. Tobias Olenyi contributed the GitHub implementation. All authors drafted the manuscript.

4.2. Journal Article: Marquet, Heinzinger *et al.*, Human Genetics (2021)

Reference: Marquet, C., Heinzinger, M., Olenyi, T., Dallago, C., Erckert, K., et al. Embeddings from protein language models predict conservation and variant effects. *Human genetics*, pages 1–19, 2021

Human Genetics
https://doi.org/10.1007/s00439-021-02411-y

ORIGINAL INVESTIGATION

Embeddings from protein language models predict conservation and variant effects

Céline Marquet[1,2] · Michael Heinzinger[1,2] · Tobias Olenyi[1,2] · Christian Dallago[1,2] · Kyra Erckert[1,2] · Michael Bernhofer[1,2] · Dmitrii Nechaev[1,2] · Burkhard Rost[1,3,4]

Received: 1 June 2021 / Accepted: 6 December 2021

Abstract

The emergence of SARS-CoV-2 variants stressed the demand for tools allowing to interpret the effect of single amino acid variants (SAVs) on protein function. While Deep Mutational Scanning (DMS) sets continue to expand our understanding of the mutational landscape of single proteins, the results continue to challenge analyses. Protein Language Models (pLMs) use the latest deep learning (DL) algorithms to leverage growing databases of protein sequences. These methods learn to predict missing or masked amino acids from the context of entire sequence regions. Here, we used pLM representations (embeddings) to predict sequence conservation and SAV effects without multiple sequence alignments (MSAs). Embeddings alone predicted residue conservation almost as accurately from single sequences as ConSeq using MSAs (two-state Matthews Correlation Coefficient—MCC—for ProtT5 embeddings of 0.596 ± 0.006 vs. 0.608 ± 0.006 for ConSeq). Inputting the conservation prediction along with BLOSUM62 substitution scores and pLM mask reconstruction probabilities into a simplistic logistic regression (LR) ensemble for Variant Effect Score Prediction without Alignments (VESPA) predicted SAV effect magnitude without any optimization on DMS data. Comparing predictions for a standard set of 39 DMS experiments to other methods (incl. ESM-1v, DeepSequence, and GEMME) revealed our approach as competitive with the state-of-the-art (SOTA) methods using MSA input. No method outperformed all others, neither consistently nor statistically significantly, independently of the performance measure applied (Spearman and Pearson correlation). Finally, we investigated binary effect predictions on DMS experiments for four human proteins. Overall, embedding-based methods have become competitive with methods relying on MSAs for SAV effect prediction at a fraction of the costs in computing/energy. Our method predicted SAV effects for the entire human proteome (~ 20 k proteins) within 40 min on one Nvidia Quadro RTX 8000. All methods and data sets are freely available for local and online execution through bioembeddings.com, https://github.com/Rostlab/VESPA, and PredictProtein.

Céline Marquet and Michael Heinzinger have contributed equally to this work.

✉ Céline Marquet
celine.marquet@tum.de
http://www.rostlab.org/

[1] Department of Informatics, Bioinformatics and Computational Biology - i12, TUM-Technical University of Munich, Boltzmannstr. 3, Garching, 85748 Munich, Germany

[2] TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany

[3] Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, Garching, 85748 Munich, Germany

[4] TUM School of Life Sciences Weihenstephan (TUM-WZW), Alte Akademie 8, Freising, Germany

Abbreviations

AI	Artificial intelligence
AUC	Area under the curve
BFD	Big Fantastic Database
CNN	Convolutional neural network
DL	Deep learning
EI	Evolutionary information
DMS	Deep mutational scanning
FNN	Feed forward neural network
GoF	Gain-of-function SAV
LoF	Loss-of-function SAV
LM	Language model
LR	Logistic regression
MAVE	Multiplexed Assays of Variant Effect
MCC	Matthews correlation coefficient
ML	Machine learning
MSA	Multiple sequence alignments

Published online: 30 December 2021

NLP	Natural language processing
OMIM	Online Mendelian Inheritance in Man
PDB	Protein Data Bank
pLM	Protein Language Model (used here: ESM-1b/1v: ProtBERT: ProtT5)
PMD	Protein mutant database
ProtT5beff	Rule-based method developed here using ProtT5 embeddings to predict binary SAV effects from single sequences
ProtT5cons	Method developed here using ProtT5 embeddings to predict residue conservation from single sequences optimizing a CNN on the unchanged pre-trained ProtT5
ReLU	Rectified linear unit
ROC	Receiver-operating characteristic
SAV	Single amino acid variant (also known as SAAV or nsSNP: or missense mutation/variant)
SOTA	State-of-the-art
SSD	Solid State Drive
SVM	Support Vector Machine
VESPA	Method developed here for Variant Effect Score Prediction without Alignments
VESPAl	Light VESPA: less accurate but faster

Introduction

Many different resources capture SAV effects. Mutations in the Spike (S) surface protein of SARS-CoV-2 have widened the attention to the complex issue of protein variant effects (Korber et al. 2020; Laha et al. 2020; Mercatelli and Giorgi 2020; O'Donoghue et al. 2020). The ability to distinguish between beneficial (= gain of function, GoF), deleterious (= loss of function, LoF) and neutral single amino acid variants (SAVs; also referred to as SAAV, missense mutations, or non-synonymous Single Nucleotide Variants: nsSNVs) continues to be a key challenge toward understanding how SAVs affect proteins (Adzhubei et al. 2010; Bromberg and Rost 2007, 2009; Ng and Henikoff 2003; Studer et al. 2013; Wang and Moult 2001). Recently, an unprecedented amount of in vitro data describing the quantitative effects of SAVs on protein function has been produced through Multiplexed Assays of Variant Effect (MAVEs), such as deep mutational scans (DMS) (Fowler and Fields 2014; Weile and Roth 2018). However, a comprehensive atlas of in vitro variant effects for the entire human proteome still remains out of reach (AVE Alliance Founding Members 2020). Yet, even for the existing experiments, intrinsic problems remain: (1) In vitro DMS data capture SAV effects upon molecular function much better than those upon biological processes, e.g., disease implications may be covered in databases such as the Online Mendelian Inheritance in Man (OMIM) (Amberger et al. 2019), but not in MaveDB (Esposito et al. 2019). (2) The vast majority of proteins have several structural domains (Liu and Rost 2003, 2004a, b); hence, most are likely to have several different molecular functions. However, each experimental assay tends to measure the impact upon only one of those functions. (3) In vivo protein function might be impacted in several ways not reproducible by in vitro assays.

Evolutionary information from MSAs is most important to predict SAV effects. Many in silico methods try to narrow the gap between known sequences and unknown SAV effects; these include (by earliest publication date): PolyPhen/PolyPhen2 (Adzhubei et al. 2010; Ramensky et al. 2002), SIFT (Ng and Henikoff 2003; Sim et al. 2012), I-Mutant (Capriotti et al. 2005), SNAP/SNAP2 (Bromberg and Rost 2007; Hecht et al. 2015), MutationTaster (Schwarz et al. 2010), Evolutionary Action (Katsonis and Lichtarge 2014), CADD (Kircher et al. 2014), PON-P2 (Niroula et al. 2015), INPS (Fariselli et al. 2015), Envision (Gray et al. 2018), DeepSequence (Riesselman et al. 2018), GEMME (Laine et al. 2019), ESM-1v (Meier et al. 2021), and methods predicting rheostat positions susceptible to gradual effects (Miller et al. 2017). Of these, only Envision and DeepSequence trained on DMS experiments. Most others trained on sparsely annotated data sets such as disease-causing SAVs from OMIM (Amberger et al. 2019), or from databases such as the protein mutant database (PMD) (Kawabata et al. 1999; Nishikawa et al. 1994). While only some methods use sophisticated algorithms from machine learning (ML; SVM, FNN) or even artificial intelligence (AI; CNN), almost all rely on evolutionary information derived from multiple sequence alignments (MSAs) to predict variant effects. The combination of evolutionary information (EI) and ML/AI has long been established as a backbone of computational biology (Rost 1996; Rost and Sander 1992, 1993), now even allowing AlphaFold2 to predict protein structure at unprecedented levels of accuracy (Jumper et al. 2021). Nevertheless, for almost no other task is EI as crucial as for SAV effect prediction (Bromberg and Rost 2007). Although different sources of input information matter, when MSAs are available, they trump all other features (Hecht et al. 2015). Even models building on the simplest EI, e.g., the BLOSUM62 matrix condensing bio-physical constraints into a 20 × 20 substitution matrix (Ng and Henikoff 2003) with no distinction between E481K (amino acid E at residue position 481 mutated to amino acid K) and E484K (part of SARS-CoV-2 Delta variant), or a simple conservation weight (Reeb et al. 2020) with no distinction of D484Q and D484K, almost reach the performance of much more complex and seemingly *advanced* methods.

Embeddings capture language of life written in proteins. Every year, algorithms improve natural language processing (NLP), in particular by feeding large text corpora into Deep Learning (DL)-based Language Models

(LMs). These advances have been transferred to protein sequences by learning to predict masked or missing amino acids using large databases of raw protein sequences as input (Alley et al. 2019; Bepler and Berger 2019a, 2021; Elnaggar et al. 2021; Heinzinger et al. 2019; Madani et al. 2020; Ofer et al. 2021; Rao et al. 2020; Rives et al. 2021). Processing the information learned by such protein LMs (pLMs), e.g., by constructing 1024-dimensional vectors of the last hidden layers, yields a representation of protein sequences referred to as embeddings [Fig. 1 in (Elnaggar et al. 2021)]. Embeddings have succeeded as exclusive input to predicting secondary structure and subcellular location at performance levels almost reaching (Alley et al. 2019; Heinzinger et al. 2019; Rives et al. 2021) or even exceeding (Elnaggar et al. 2021; Littmann et al. 2021c; Stärk et al. 2021) state-of-the-art (SOTA) methods using EI from MSAs as input. Embeddings even succeed in substituting sequence similarity for homology-based annotation transfer (Littmann et al. 2021a, b) and in predicting the effect of mutations on protein–protein interactions (Zhou et al. 2020). The power of such embeddings has been increasing with the advance of algorithms (Bepler and Berger 2021; Elnaggar et al. 2021; Rives et al. 2021). ESM-1v demonstrated pre-trained pLMs predicting SAV effect without any supervision at state-of-the-art level on DMS data using solely mask reconstruction probabilities (Meier et al. 2021). Naturally, there will be some limit to such improvements. However, the advances over the last months prove that this limit had not been reached by the end of 2020.

Here, we analyzed ways of using embeddings from pre-trained pLMs to predict the effect of SAVs upon protein function with a focus on molecular function, using experimental data from DMS (Esposito et al. 2019) and PMD (Kawabata et al. 1999). The embeddings from the pre-trained pLMs were not altered or optimized to suit the subsequent 2nd step of supervised training on data sets with more limited annotations. In particular, we assessed two separate supervised prediction tasks: conservation and SAV effects. First, we utilized pre-trained pLMs (ProtBert, ProtT5, ESM-1b) as static feature encoders (without fine-tuning the pLMs) to derive input embeddings for developing a method predicting the conservation that we could read off a family of aligned sequences (MSA) for each residue without actually generating the MSA. Second, we trained a Logistic Regression (LR) ensemble to predict SAV effect using (2a) the predictions of the best conservation predictor (ProtT5cons) together with (2b) substitution scores of BLOSUM62 and (2c) substitution probabilities of the pLM ProtT5. While substitution probabilities alone already correlated with DMS scores, we observed that adding conservation predictions together with BLOSUM62 increased performance. The resulting model for Variant Effect Score Prediction without Alignments (VESPA) was competitive with more complex solutions in terms of correlation with experimental DMS scores and computational and environmental costs. Additionally, for a small drop in prediction performance, we created a computationally more efficient method, dubbed VESPA-light (or short: VESPAl), by excluding substitution probabilities to allow proteome-wide analysis to complete after the coffee break on a single machine (40 min for human proteome on one Nvidia Quadro RTX 8000).

Methods

Data sets

In total, we used five different datasets. *ConSurf10k* was used to train and evaluate a model on residue conservation prediction. *Eff10k* was used to train SAV effect prediction. *PMD4k* and *DMS4* were used as test sets to assess the prediction of binary SAV effects. The prediction of continuous effect scores was evaluated on *DMS39*.

***ConSurf10k* assessed conservation.** The method predicting residue conservation used *ConSurf-DB* (Ben Chorin et al. 2020). This resource provided sequences and conservation for 89,673 proteins. For all, experimental high-resolution three-dimensional (3D) structures were available in the Protein Data Bank (PDB) (Berman et al. 2000). As standard-of-truth for the conservation prediction, we used the values from ConSurf-DB generated using HMMER (Mistry et al. 2013), CD-HIT (Fu et al. 2012), and MAFFT-LINSi (Katoh and Standley 2013) to align proteins in the PDB (Burley et al. 2019). For proteins from families with over 50 proteins in the resulting MSA, an evolutionary rate at each residue position is computed and used along with the MSA to reconstruct a phylogenetic tree. The ConSurf-DB conservation scores ranged from 1 (most variable) to 9 (most conserved). The PISCES server (Wang and Dunbrack 2003) was used to redundancy reduce the data set, such that no pair of proteins had more than 25% pairwise sequence identity. We removed proteins with resolutions > 2.5 Å, those shorter than 40 residues, and those longer than 10,000 residues. The resulting data set (ConSurf10k) with 10,507 proteins (or domains) was randomly partitioned into training (9392 sequences), cross-training/validation (555), and test (519) sets.

***Eff10k* assessed SAV effects.** This dataset was taken from the SNAP2 development set (Hecht et al. 2015). It contained 100,737 binary SAV-effect annotations (neutral: 39,700, effect: 61,037) from 9594 proteins. The set was used to train an ensemble method for SAV effect prediction. For this, we replicated the cross-validation (CV) splits used to develop SNAP2 by enforcing that clusters of sequence-similar proteins were put into the same CV split. More specifically, we clustered all sequence-similar proteins (PSI-BLAST *E* value < 1e-3) using single-linkage clustering, i.e., all connected nodes (proteins) were put into the same cluster. By placing all proteins within one cluster into the same CV split and rotating the splits, such that every split was used exactly once for testing, we ascertained that no pair of proteins between train and test shared significant sequence similarity (PIDE). More details on the dataset are given in SNAP2 (Hecht et al. 2015).

***PMD4k* assessed binary SAV effects.** From Eff10k, we extracted annotations that were originally adopted from PMD ("no change" as "neutral"; annotations with any level of increase or decrease in function as "effect"). This yielded 51,817 binary annotated SAVs (neutral: 13,638, effect: 38,179) in 4061 proteins. PMD4k was exclusively used for testing. While these annotations were part of Eff10k, all performance estimates for PMD4k were reported only for the PMD annotations in the testing subsets of the cross-validation splits. As every protein in Eff10k (and PMD4k) was used exactly once for testing, we could ascertain that there was no significant (prediction by homology-based inference possible) sequence-similarity between PMD4k and our training splits.

***DMS4* sampled large-scale DMS in vitro experiments annotating binary SAV effects.** This set contained binary classifications (effect/non-effect) for four human proteins (corresponding genes: BRAC1, PTEN, TPMT, PPARG) generated previously (Reeb 2020). These were selected as they were the first proteins with comprehensive DMS experiments including synonymous variants (needed to map from continuous effect scores to binary effect vs. neutral) resulting in 15,621 SAV annotations (Findlay et al. 2018; Majithia et al. 2016; Matreyek et al. 2018). SAVs with beneficial effect (= gain of function) were excluded, because they disagree between experiments (Reeb et al. 2020). The continuous effect scores of the four DMS experiments were mapped to binary values (effect/neutral) by considering the 95% interval around the mean of all experimental measurements as neutral, and the 5% tails of the distribution as "effect", as described in more detail elsewhere (Reeb et al. 2020). In total, the set had 11,788 neutral SAVs and 3516 deleterious effect SAVs. Additionally, we used two other thresholds: the 90% interval from mean (8926 neutral vs. 4545 effect) and the 99% interval from mean (13,506 neutral vs. 1,548 SAVs effect).

***DMS39* collected DMS experiments annotating continuous SAV effects.** This set was used to assess whether the methods introduced here, although trained only on binary effect data from Eff10k, had captured continuous effect scales as measured by DMS. The set was a subset of 43 DMS experiments assembled for the development of DeepSequence (Riesselman et al. 2018). From the original compilation, we excluded an experiment on tRNA as it is not a protein, on the toxin–antitoxin complex as it comprises multiple proteins and removed experiments for which only double variants existed. DMS39 contained 135,665 SAV scores, in total. The number of SAVs per experiment varied substantially between the 39 with an average of 3625 SAVs/experiment, a median of 1962, a minimum of 21, and a maximum of 12,729. However, to avoid any additional biases in the comparison to other methods, we avoided any further filtering step.

Input features

For the prediction of residue conservation, all newly developed methods exclusively trained on embeddings from pre-trained pLMs without fine-tuning those (no gradient was backpropagated to the pLM). The predictions of the best-performing method for conservation prediction were used in a second step together with substitution scores from BLOSUM62 and substitution probabilities from ProtT5 as input features to predict binary SAV effects.

Embeddings from pLMs: For conservation prediction, we used embeddings from the following pLMs: *ProtBert* (Elnaggar et al. 2021) based on the NLP (Natural Language Processing) algorithm BERT (Devlin et al. 2019) trained on Big Fantastic Database (BFD) with over 2.3 million protein sequences (Steinegger and Söding 2018), ESM-1b (Rives et al. 2021) that is conceptually similar to (Prot)BERT (both use a Transformer encoder) but trained on UniRef50 (The UniProt Consortium 2021) and *ProtT5-XL-U50* (Elnaggar et al. 2021) (for simplicity referred to as *ProtT5*) based on the NLP sequence-to-sequence model T5 (transformer encoder–decoder architecture) (Raffel et al. 2020) trained on BFD and fine-tuned on Uniref50. All embeddings were obtained from the bio_embeddings pipeline (Dallago et al. 2021). As described in ProtTrans, only the encoder side of ProtT5 was used and embeddings were extracted in half-precision (Elnaggar et al. 2021). The per-residue embeddings were extracted from the last hidden layer of the models with size 1024 × L (1280 for ESM-1b), where L is the length of the protein sequence and 1024 (or 1280 for ESM-1b) is the dimension of the hidden states/embedding space of ESM-1b, ProtBert, and ProtT5.

Context-dependent substitution probabilities: The training objective of most pLMs is to reconstruct corrupted amino acids from their non-corrupted protein sequence context. Repeating this task on billions of sequences allows pLMs to learn a probability of how likely it is to observe a token (an amino acid) at a certain position in the protein sequence. After pre-training, those probabilities can be extracted from pLMs by masking/corrupting one token/amino acid at a time, letting the model reconstruct it based on non-corrupted sequence context and repeating this for each token/amino acid in the sequence. For each protein, this gives a vector of length L by 20 with L being the protein's

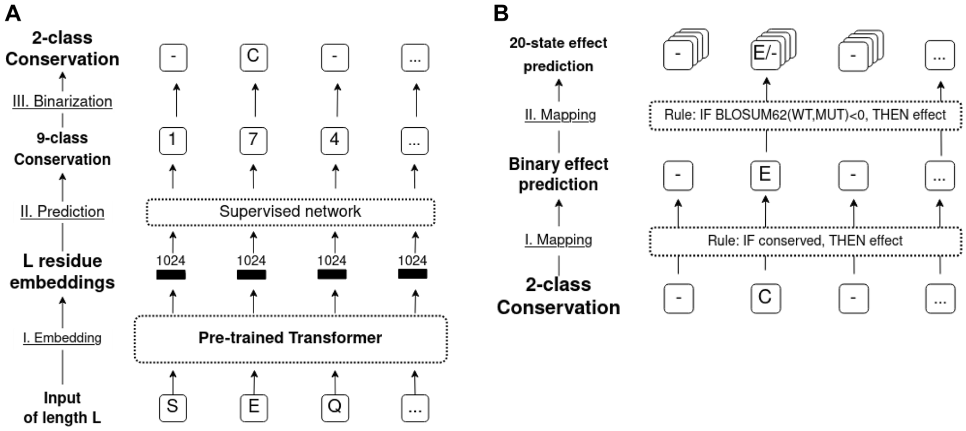

Fig. 1 Sketch of methods. Panel A sketches the conservation prediction pipeline: (I) embed protein sequence ("SEQ") using a pLM [here: ProtBERT, ProtT5 (Elnaggar et al. 2021) or ESM-1b (Meier et al. 2021)]. (II) Input embedding into supervised method (here: logistic regression, FNN or CNN) to predict conservation in 9-classes as defined by ConSurf-DB (Ben Chorin et al. 2020). (III) Map nine-class predictions > 5 to *conserved* (C), others to *non-conserved* (–). Panel B shows the use of binary conservation predictions as input to SAV effect prediction by (I) considering all residue positions predicted as conserved (C) as effect (E), all others as neutral (ProtT5cons-19equal and ConSeq-19equal). (II) Residues predicted as conserved are further split into specific substitutions (SAVs) predicted to have an effect (E) or not (–) if the corresponding BLOSUM62 score is < 0, all others are predicted as neutral (ProtT5-beff, ConSeq-BLOSUM62)

length and 20 being the probability distribution over the 20 standard amino acids. It was shown recently (Meier et al. 2021) that these probabilities provide a context-aware estimate for the effect of SAVs, i.e., the reconstruction probabilities depend on the protein sequence, and other methods have made use of similar probabilities (Hopf et al. 2017; Riesselman et al. 2018). To generate input features for our SAV effect predictor, we used, as suggested by Meier et al. (2021), the log-odds ratio between the probability of observing the wild-type amino acid at a certain position and the probability of observing a specific mutant at the same position: $\log\left(p\left(X_{i,mutant}\right)\right) - \log\left(p\left(X_{i,wildtype}\right)\right)$. The term $p\left(X_{i,mutant}\right)$ described the probability of an SAV occurring at position i and $p\left(X_{i,wildtype}\right)$ described the corresponding probability of the wild-type occurrence (native amino acid). To extract these probabilities for SAV effect prediction, we only considered the pLM embeddings correlating best with conservation (ProtT5). Additionally, we extracted probabilities for ProtBert on ConSurf10k to analyze in more detail the mistakes that ProtBert makes during reconstruction (SOM Fig. S5, S6).

Context-independent BLOSUM62 substitution scores: The BLOSUM substitution matrix gives a log-odds ratio for observing an amino acid substitution irrespective of its position in the protein (Henikoff and Henikoff 1992), i.e., the substitution score will not depend on a specific protein or the position of a residue within a protein but rather focuses on bio-chemical and bio-physical properties of amino acids. Substitution scores in BLOSUM were derived from comparing the log-odds of amino acid substitutions among well-conserved protein families. Typically applied to align proteins, BLOSUM scores are also predictive of SAV effects (Ng and Henikoff 2003; Sruthi et al. 2020).

Method development

In our three-stage development, we first compared different combinations of network architectures and pLM embeddings to predict residue conservation. Next, we combined the best conservation prediction method with BLOSUM62 substitution scores to develop a simple rule-based prediction of binary SAV effects. In the third step, we combined the predicted conservation, BLOSUM62, and substitution probabilities to train a new method predicting SAV effects for binary data from Eff10k and applied this method to non-binary DMS data.

Conservation prediction (ProtT5cons, Fig. 1A): Using either ESM-1b, ProtBert, or ProtT5 embeddings as input (Fig. 1a), we trained three supervised classifiers to distinguish between nine *conservation classes* taken from ConSurf-DB (early stop when optimum reached for ConSurf10k validation set). The objective of this task was to learn the prediction of family conservation from ConSurf-DB (Ben Chorin et al. 2020) based on the nine conservation classes introduced by that method that range from 1 (variable) to 9 (conserved) for each residue in a protein, i.e., this task implied a multi-class per-residue prediction. Cross-entropy loss together with Adam (Kingma and Ba 2014) was used to optimize each network toward predicting one out of nine conservation classes for each residue in a protein (per-token/per-residue task).

The models were: (1) standard Logistic Regression (LR) with 9000 (9 k) free parameters; (2) feed-forward neural network (FNN; with two FNN layers connected through the so-called ReLU (rectified linear unit) activations (Fukushima 1969); dropout rate 0.25; 33 k free parameters); (3) standard convolutional neural network (CNN; with two convolutional layers with a window size of 7, connected through ReLU activations; dropout rate of 0.25; 231 k free parameters). To put the number of free parameters into perspective: the ConSurf10k data set contained about 2.7 million samples, i.e., an order of magnitude more samples than free parameters of the largest model. On top of the 9-class prediction, we implemented a binary classifier (*conserved/non-conserved*; threshold for projecting nine to two classes optimized on validation set). The best-performing model (CNN trained on ProtT5) was referred to as ProtT5cons.

Rule-based binary SAV effect prediction (ProtT5beff, Fig. 1B): For rule-based binary SAV effect (*effect/neutral*) prediction, we considered multiple approaches. The first and simplest approach was to introduce a threshold to the output of ProtT5cons (no optimization on SAV data). Here, we marked all residues predicted to be conserved (conservation score > 5) as "effect"; all others as "neutral". This first level treated all 19 non-native SAVs at one sequence position equally (referred to as "19equal" in tables and figures). To refine, we followed the lead of SIFT (Ng and Henikoff 2003) using the BLOSUM62 (Henikoff and Henikoff 1992) substitution scores. This led to the second rule-based method dubbed *BLOSUM62bin* which can be considered a naïve baseline: SAVs less likely than expected (negative values in BLOSUM62) were classified as "effect"; all others as "neutral". Next, we combined both rule-based classifiers to the third rule-based method, dubbed *ProtT5beff* ("effect" if ProtT5cons predicts conserved, i.e., value > 5, and BLOSUM62 negative, otherwise "neutral", Fig. 1b). This method predicted binary classifications (effect/neutral) of SAVs without using any experimental data on SAV effects for optimization by merging position-aware information from ProtT5cons and variant-aware information from BLOSUM62.

Supervised prediction of SAV effect scores (*VESPA* and *VESPAl)*: For variant effect score prediction without alignments (VESPA), we trained a balanced logistic regression (LR) ensemble method as implemented in SciKit (Pedregosa et al. 2011) on the cross-validation splits of Eff10k. We rotated the ten splits of Eff10k, such that each data split was used exactly once for testing, while all remaining splits were used for training. This resulted in ten individual LRs trained on separate datasets. All of those were forced to share the same hyper-parameters. The hyper-parameters that differed from SciKit's defaults were: (1) *balanced weights*: class weights were inversely proportional to class frequency in input data; (2) *maximum number of iterations taken for the solvers to converge* was set to 600. The learning objective of each was to predict the probability of binary class membership (effect/neutral). By averaging their output, we combined the ten LRs to an ensemble method: $VESPA = ensemble\ of\ LRs = \frac{1}{10}\sum_{i=1}^{10} LR_i$. The output of VESPA is bound to [0,1] and by introducing a threshold can be readily interpreted as a probability for an SAV to be "neutral" (VESPA < 0.5) or to have "effect" (VESPA ≥ 0.5). As input for VESPA, we used 11 features to derive one score for each SAV; nine were the position-specific conservation probabilities predicted by ProtT5cons; one was the variant-specific substitution score from BLOSUM62, the other the variant- and position-specific log-odds ratio of ProtT5's substitution probabilities. To reduce the computational costs of VESPA, we introduced the "light" version VESPAl using only conservation probabilities and BLOSUM62 as input and thereby circumventing the computationally more costly extraction of the log-odds ratio. Both VESPA and VESPAl were only optimized on binary effect data from Eff10k and never encountered continuous effect scores from DMS experiments during any optimization.

Evaluation

Conservation prediction—ProtT5cons: To put the performance of ProtT5cons into perspective, we generated ConSeq (Berezin et al. 2004) estimates for conservation through PredictProtein (Bernhofer et al. 2021) using MMseqs2 (Steinegger and Söding 2018) and PSI-BLAST (Altschul et al. 1997) to generate MSAs. These were "estimates" as opposed to the standard-of-truth from ConSurf-DB, because, although they actually generated entire MSAs, the method for MSA generation was "just" MMseqs2 as opposed to HMMER (Mistry et al. 2013), and MAFFT-LINSi (Katoh and Standley 2013) for ConSurf-DB and the computation of weights from the MSA also required less computing resources. A random baseline resulted from randomly shuffling ConSurf-DB values.

Binary effect prediction—ProtT5beff: To analyze the performance of VESPA and VESPAl, we compared results to SNAP2 (Hecht et al. 2015) at the default binary threshold (score > − 0.05, default value suggested in original publication) on PMD4k and DMS4. Furthermore, we evaluated the rule-based binary SAV effect prediction ProtT5beff on the same datasets. To assess to which extent performance of ProtT5beff could be attributed to mistakes in ProtT5cons, we replaced residue conservation from ProtT5cons with conservation scores from ConSeq and applied the same two rule-based approaches as explained above (*ConSeq 19equal*: conserved predictions at one sequence position were considered "effect" for all 19 non-native SAVs and *ConSeq blosum62*: only negative BLOSUM62 scores at residues predicted as conserved were considered "effect"; all others considered "neutral" with both using the same threshold in conservation as for our method, i.e., conservation > 5 for effect) for PMD4k and DMS4. This failed for 122 proteins on PMD4k (3% of PMD4k), because the MSAs were deemed too small. We also compared ProtT5beff to the baseline based only on BLOSUM62 with the same thresholds as above (BLOSUM62bin). Furthermore, we compared to SNAP2 at default binary threshold of effect: SNAP2 score > − 0.05 (default value suggested in original publication). SNAP2 failed for four of the PMD4k proteins (0.1% of PMD4k). For the random baseline, we randomly shuffled ground truth values for each PMD4k and DMS4.

Continuous effect prediction—VESPA: We evaluated the performance of VESPA and VESPAl on DMS39 comparing to MSA-based DeepSequence (Riesselman et al. 2018) and GEMME (Laine et al. 2019), and the pLM-based ESM-1v (Meier et al. 2021). Furthermore, we evaluated log-odds ratios from ProtT5's substitution probabilities and BLOSUM62 substitution scores as a baseline. The DeepSequence predictions were copied from the supplement to the original publication (Riesselman et al. 2018), GEMME correlation coefficients were provided by the authors, and ESM-1v predictions were replicated using the online repository of ESM-1v. We used the publicly available ESM-1v scripts to retrieve "*masked-marginals*" for each of the five ESM-1v models and averaged over their outputs, because this strategy gave best performance according to the authors. If a protein was longer than 1022 (the maximum sequence length that ESM-1v can process), we split the sequence into non-overlapping chunks of length 1022. VESPA, VESPAl, and ESM-1v predictions did not use MSAs and therefore provided results for the entire input sequences, while DeepSequence and GEMME were limited to residues to which enough other protein residues were aligned in the MSAs.

Performance measures: We applied the following standard performance measures:

$$\text{Q2} = 100 \cdot \frac{\text{(Number of residues predicted correctly in 2 states)}}{\text{(Number of all residues)}}. \tag{1}$$

Q2 scores (Eq. 1) described both binary predictions (conservation and SAV effect). The same held for F1-scores (Eq. 6, 7) and MCC (Matthews Correlation Coefficient, Eq. 8). We defined conserved/effect as the positive class and non-conserved/neutral as the negative class (indices "+" for positive, "–" for negative) and used the standard abbreviations of TP (true positives: number of residues predicted and observed as conserved/effect), TN (true negatives: predicted and observed as non-conserved/neutral), FP (false positives: predicted conserved/effect, observed non-conserved/neutral), and FN (false negatives: predicted non-conserved/neutral, observed conserved/effect)

$$Accuracy_+ = Precision_+ = Positive\,Predicted\,Value = \frac{TP}{TP + FP} \tag{2}$$

$$Accuracy\_ = Precision\_ = NegativePredictedValue = \frac{TN}{TN + FN} \tag{3}$$

$$Coverage_+ = Recall_+ = Sensitivity = \frac{TP}{TP + FN} \tag{4}$$

$$Coverage\_ = Recall\_ = Specificity = \frac{TN}{TN + FP} \tag{5}$$

$$F1_+ = 100 \bullet 2 \bullet \frac{Precision_+ \bullet Recall_+}{Precision_+ + Recall_+} \tag{6}$$

$$F1\_ = 100 \bullet 2 \bullet \frac{Precision\_ \bullet Recall\_}{Precision\_ + Recall\_} \tag{7}$$

$$MCC = \frac{TP \bullet TN - FP \bullet FN}{\sqrt{(TP + FP) \bullet (TP + FN) \bullet (TN + FP) \bullet (TN + FN)}} \tag{8}$$

$$Q9 = 100 \bullet \frac{Number\,of\,residues\,predicted\,correctly\,in\,9\,states}{Number\,of\,all\,residues}. \tag{9}$$

Q9 is exclusively used to measure performance for the prediction of nine classes of conservation taken from ConSurf-DB. Furthermore, we considered the Pearson correlation coefficient

$$r_P = \rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y}, \tag{10}$$

and the Spearman correlation coefficient where raw scores (X, Y of Eq. 10) are converted to ranks

$$r_S = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{Xrg_X} \sigma_{rg_Y}} \tag{11}$$

for continuous effect prediction.

Error estimates: We estimated symmetric 95% confidence intervals (CI Eq. 12) for all metrics using bootstrapping (Efron et al. 1996) by computing 1.96* standard deviation (SD) of randomly selected variants from all test sets with replacement over $n = 1000$ bootstrap sets

$$CI = 1.96 \bullet SD = 1.96 \bullet \sqrt{\frac{\sum (y_i - \bar{y})^2}{n}}, \tag{12}$$

with y_i being the metric for each bootstrap sample and $\bar{y}$ the mean over all bootstrap samples. We considered differences in performance significant if two CIs did not overlap.

Probability entropy: To investigate the correlation between embeddings and conservation classes of ConSurf-DB, we computed the entropy of pLM substitution probabilities (p) as

$$Entropy(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i. \tag{13}$$

Results

We first showed that probabilities derived from pLMs sufficed for the prediction of residue conservation from pLM embeddings without using MSAs (data set *ConSurf10k*; method *ProtT5cons*). Next, we presented a non-parametric rule-based SAV effect prediction based on predicted conservation (IF “predicted conserved” THEN “predict effect”; method *ProtT5beff*). We refined the rule-based system through logistic regression (LR) to predict SAV effect on variants labeled with “effect” or “neutral” (data set *Eff10k*; methods *VESPA, VESPAl*). Finally, we established that these new methods trained on binary data (effect/neutral) from *Eff10k* correlated with continuous DMS experiments.

Embeddings predicted conservation: First, we established that protein Language Models (pLMs) capture information correlated with residue conservation without ever seeing any such labels. As a standard-of-truth, we extracted the categorical conservation scores ranging from 1 to 9 (9: highly conserved, 1: highly variable) from ConSurf-DB (Ben Chorin et al. 2020) for a non-redundant subset of proteins with experimentally known structures (data set ConSurf10k). Those conservation scores correlated with the mask reconstruction probabilities output by ProtBert (Fig. 2). More specifically, one amino acid was corrupted at a time and ProtBert reconstructed it from non-corrupted sequence context. For instance, when corrupting and reconstructing all residues in ConSurf10k (one residue at a time), ProtBert assigned a probability to the native and to each of the 19 non-native (SAVs) amino acids for each position in the protein. Using those “*substitution probabilities*”, ProtBert correctly predicted the native amino acid in 45.3% of all cases compared to 9.4% for a random prediction of the most frequent amino acid (Fig. S4). The entropy of these probability distributions correlated slightly with conservation (Fig. 2, Spearman’s R = -–0.374) although never trained on such labels.

Next, we established that residue conservation can be predicted directly from embeddings by training a supervised network on data from ConSurf-DB. We exclusively used embeddings of pre-trained pLMs (ProtT5, ProtBert (Elnaggar et al. 2021), ESM-1b (Rives et al. 2021)), as input to relatively simple machine learning models (Fig. 1). Even the simplistic logistic regression (LR) reached levels of performance within about 20% of ConSeq (Berezin et al. 2004) conservation scores, which were derived from MSAs generated by the fast alignment method MMseqs2 (Steinegger and Söding 2017) (Fig. 3). The top prediction used ProtT5 embeddings which consistently outperformed predictions from ESM-1b and ProtBERT embeddings. For all three types of embeddings, the CNN outperformed the FNN, and these outperformed the LR. Differences between ProtBert and ProtT5 were statistically significant (at the 95% confidence interval, Eq. 12), while improvements from ProtT5 over ESM-1b were mostly insignificant. The ranking of the embeddings and models remained stable across several performance measures ($F1_{effect}$, $F1_{neutral}$, MCC, Pearson correlation coefficient, Table S1).

ConSurf-DB (Ben Chorin et al. 2020) simplifies family conservation to a single digit integer (9: highly conserved, 1: highly variable). We further reduced these classes to a binary classification (conserved/non-conserved) to later

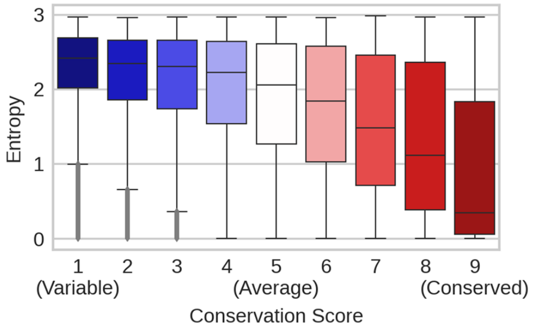

Fig. 2 pLMs captured conservation without supervised training or MSAs. ProtBert was optimized to reconstruct corrupted input tokens from non-corrupted sequence context (masked language modeling). Here, we corrupted and reconstructed all proteins in the ConSurf10k dataset, one residue at a time. For each residue position, ProtBert returned the probability for observing each of the 20 amino acids at that position. The higher one probability (and the lower the corresponding entropy), the more certain the pLM predicts the corresponding amino acid at this position from non-corrupted sequence context. Within the displayed boxplots, medians are depicted as black horizontal bars; whiskers are drawn at the 1.5 interquartile range. The x-axis gives categorical conservation scores (1: highly variable, 9: highly conserved) computed by ConSurf-DB (Ben Chorin et al. 2020) from multiple sequence alignments (MSAs); the y-axis gives the probability entropy (Eq. 13) computed without MSAs. The two were inversely proportional with a Spearman’s correlation of -0.374 (Eq. 11), i.e., the more certain ProtBert’s prediction, the lower the entropy and the higher the conservation for a certain residue position. Apparently, ProtBert had extracted information correlated with residue conservation during pre-training without having ever seen MSAs or any labeled data

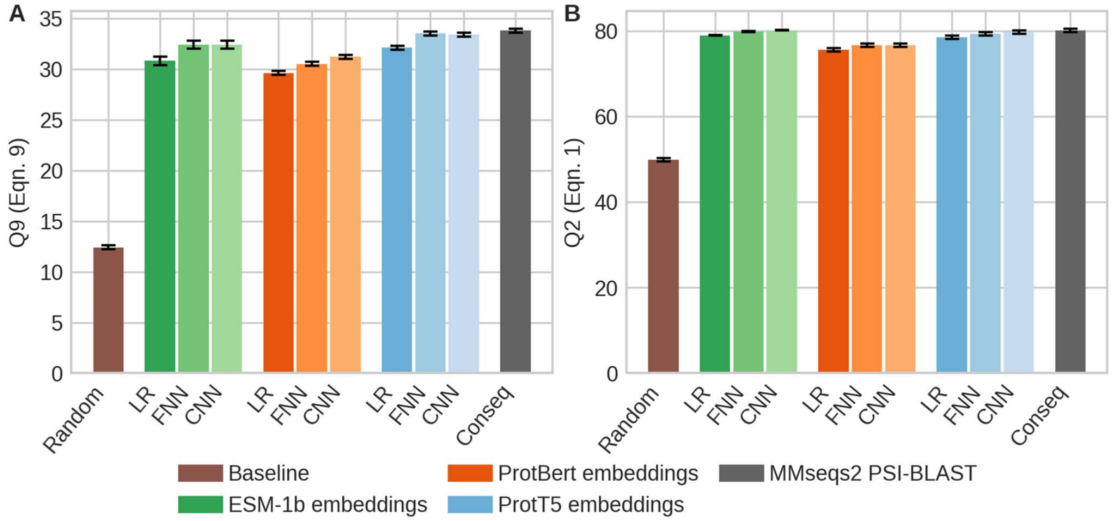

Fig. 3 Conservation predicted accurately from embeddings. Data: hold-out test set of *ConSurf10k* (519 sequences); panel A: nine-state per-residue accuracy (Q9, Eq. 9) in predicting conservation as defined by ConSurf-DB (Ben Chorin et al. 2020); panel B: two-state per-residue accuracy (Q2, Eq. 1; conservation score > 5: conserved, non-conserved otherwise). Supervised models (trained on *ConSurf10k*): **LR**: logistic regression (9,000 = 9 k free parameters), *FNN* feed-forward network (33 k parameters), and *CNN* convolutional neural network (231 k parameters with 0.25 dropout rate); methods: *ConSeq* computation of conservation weight through multiple sequence alignments (MSAs) (Berezin et al. 2004); *Random* random label swap. Model inputs were differentiated by color (green: ESM-1b embeddings (Rives et al. 2021), red: ProtBert embeddings (Elnaggar et al. 2021), blue: ProtT5 embeddings (Elnaggar et al. 2021), gray: MSAs (MMseqs2 (Steinegger and Söding 2017), and PSI-BLAST (Altschul et al. 1997)). Black whiskers mark the 95% confidence interval (± 1.96 SD; Eq. 12). ESM-1b and ProtT5 embeddings outperformed those from ProtBERT (Elnaggar et al. 2021); differences between ESM-1b and ProtT5 were not statistically significant, but ProtT5 consistently outperformed ESM-1b in all metrics but Q2 (Table S1). ESM-1b and ProtT5 as input to the CNN came closest to ConSeq (Table S1)

transfer information from conservation to binary SAV effect (effect/neutral) more readily. The optimal threshold for a binary conservation prediction was 5 (> 5 conserved, Fig. S1). However, performance was stable across a wide range of choices: between values from 4 to 7, MCC (Eq. 8) changed between 0.60 and 0.58, i.e., performance varied by 3.3% for 44.4% of all possible thresholds (Fig. S1). This was explained by the nine- and two-class confusion matrices (Fig. S2 and S3) for *ProtT5cons*, which showed that most mistakes were made between neighboring classes of similar conservation, or between the least conserved classes 1 and 2.

Conservation-based prediction of binary SAV effect better for DMS4 than for PMD4k? Next, we established that we could use the predicted conservation of ProtT5cons for rule-based binary SAV effect prediction without any further optimization and without any MSA. In using predicted conservation to proxy SAV effect, we chose the method best in conservation prediction, namely the CNN using ProtT5 embeddings (method dubbed *ProtT5cons*, Fig. 1B). The over-simplistic approach of considering any residue predicted as conserved to have an effect irrespective of the SAV (meaning: treat all 19 non-native SAVs alike) was referred to as "19equal". We refined this rule-based approach by combining conservation prediction with a binary BLOSUM62 score (effect: if ProtT5cons predicted conserved AND BLOSUM62 < 0, neutral otherwise), which we referred to as *ProtT5beff*. For PMD4k, the following results were common to all measures reflecting aspects of precision and recall through a single number ($F1_{effect}$, $F1_{neutral}$ and MCC). First, the expert method SNAP2 trained on Eff10k (superset of PMD4k) achieved numerically higher values than all rule-based methods introduced here. Second, using the same SAV effect prediction for all 19 non-native SAVs consistently reached higher values than using the BLOSUM62 values (Fig. 4 and Table 1: *19equal* higher than *blosum62*). For some measures (Q2, $F1_{effect}$), values obtained using ConSeq for conservation (i.e., a method using MSAs) were higher than those for the ProtT5cons prediction (without using MSAs), while for others (MCC, $F1_{neutral}$), this was reversed (Fig. 4, Table 1, Table S2).

Most performances differed substantially between PMD4k and DMS4, i.e., the first four proteins (BRAC1, PTEN, TPMT, and PPARG) for which we had obtained large-scale experimental DMS measures that could be converted into a binary scale (effect/neutral). First, using BLOSUM62 to convert ProtT5cons into SAV-specific predictions

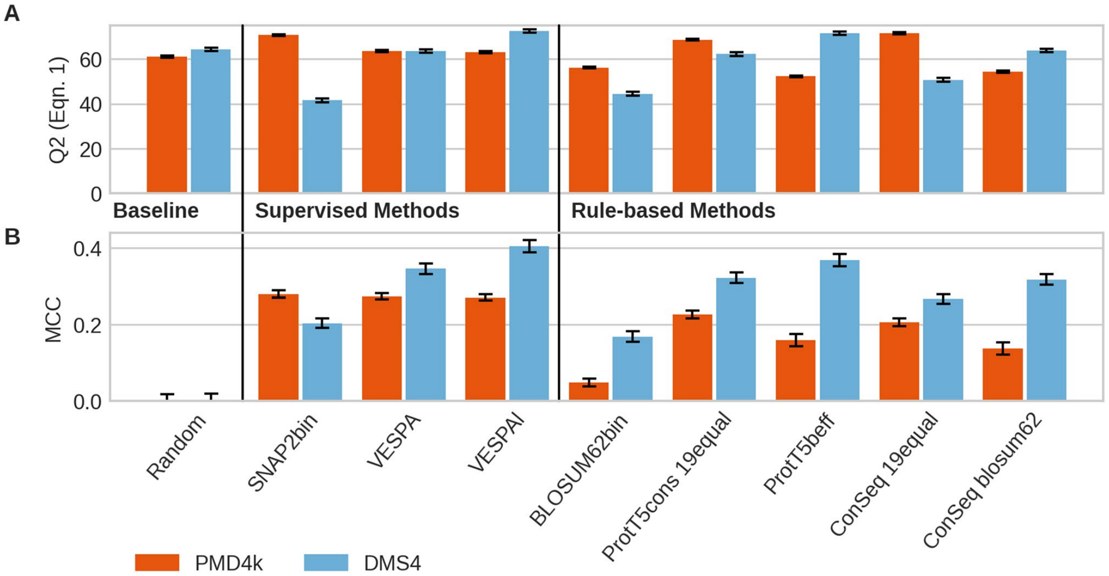

Fig. 4 Embedding-based binary SAV effect prediction is seemingly competitive. Data: *PMD4k* (red bars; 4 k proteins from PMD (Kawabata et al. 1999)); *DMS4* (blue bars) first four human proteins (BRAC1, PTEN, TPMT, PPARG) with comprehensive experimental DMS measurements including synonyms (here 95% threshold) (Reeb et al. 2020). Methods: SUPERVISED: **a** *SNAP2bin*: effect SNAP2 score > − 0.05, otherwise neutral; **b** *VESPA*: effect VESPA score > =0.5, otherwise neutral; **c** *VESPAl*: effect VESPAl score > =0.5, otherwise neutral. RULE-BASED: **d** *BLOSUM62bin*: irrespective of residue position, negative BLOSUM62 scores predicted as effect, others as neutral; **e** *ProtT5cons|ConSeq 19equal*: all 19 non-native SAVs predicted equally: effect if ProtT5cons|ConSeq predicted residue position to be conserved, otherwise neutral; **f** *ProtT5beff|ConSeq blosum62*: effect if ProtT5cons|ConSeq predicts conserved and BLOSUM62 negative, otherwise neutral. BASELINE: **g** *Random*: random shuffle of experimental labels. All values for DMS4 computed for binary (effect/neutral) mapping of experimental DMS values with panel A giving the two-state per-residue accuracy (Q2, Eq. 1) and panel B giving the Matthews Correlation Coefficient (MCC, Eq. 8). Error bars: Black bars mark the 95% confidence interval (±1.96 SD, Eq. 12). For all methods, the MCC differences between the two data sets PMD4k and DMS4 were statistically significant (exception: random)

outperformed the MSA-based conservation lookup from ConSeq, the expert method SNAP2 trained on PMD4k (Table 1: ProtT5beff highest rule-based), and the newly introduced VESPA. Second, combining the BLOSUM62 matrix with conservation also improved ConSeq (Table 1: ConSeq: *19equal* lower than *blosum62*). Third, ranking across different performance measures correlated much better than for PMD4k (Tables S1–S5). As the mapping from continuous DMS effect scores to binary labels might introduce systematic noise, we also investigated different thresholds for this mapping. However, results for DMS4 at intervals of 90% (Table S3) and 99% (Table S5) around the mean showed similar trends.

We trained a logistic regression (LR) ensemble (VESPA) on cross-validation splits replicated from the SNAP2 development set. For binary effect prediction, we introduced a threshold (≥ 0.5 effect, otherwise neutral) to the output scores of VESPA. When comparing VESPA and VESPAl (light version of VESPA) to the other methods on PMD4k, we observed a different picture than for the rule-based approaches. While SNAP2 still resulted in the highest MCC (0.28 ± 0.01), it was not significantly higher than that of VESPA and VESPAl (MCC: 0.274 ± 0.09 and 0.271 ± 0.09, respectively), and its development set overlapped with PMD4k. When evaluating the methods on DMS4, the best-performing method, VESPAl (MCC 0.405 ± 0.016), outperformed SNAP2 (MCC 0.204 ± 0.012) and VESPA (MCC 0.346 ± 0.014) as well as all rule-based methods (Table 1). We observed the same trends for other intervals (Tables S3–S5).

pLMs predicted SAV effect scores without MSAs. Could VESPA, trained on binary effect data (Eff10k) capture continuous SAV effect scores measured by DMS? For ease of comparison with other methods, we chose all 39 DMS experiments (DMS39) with single SAV effect data assembled for the development of DeepSequence (Riesselman et al. 2018). Several methods have recently been optimized on DMS data, e.g., the apparent state-of-art (SOTA), DeepSequence trained on the MSAs of each of those 39 experiments. Another recent method using evolutionary

Table 1 Performance in binary SAV effect prediction[a]

Data set	PMD4k		DMS4	
Method/metric	Q2 (Eq. 1)	MCC (Eq. 8)	Q2 (Eq. 1)	MCC (Eq. 8)
Random	61.08% ± 0.41	− 0.002 ± 0.016	64.27% ± 0.76	− 0.001 ± 0.018
Supervised methods				
SNAP2bin	70.66% ± 0.39	0.280 ± 0.010	41.55% ± 0.82	0.204 ± 0.012
VESPA	63.52% ± 0.43	0.274 ± 0.086	63.56% ± 0.79	0.346 ± 0.014
VESPAl	63.04% ± 0.43	0.271 ± 0.085	72.59% ± 0.72	**0.405 ± 0.016**
Rule-based methods				
BLOSUM62bin	56.17% ± 0.43	0.049 ± 0.010	44.47% ± 0.84	0.169 ± 0.014
ProtT5cons-19equal	68.58% ± 0.41	0.227 ± 0.010	62.20% ± 0.82	0.322 ± 0.014
ProtT5-beff	52.26% ± 0.43	0.160 ± 0.016	71.47% ± 0.75	0.369 ± 0.016
ConSeq-19equal	**71.51% ± 0.39**	0.206 ± 0.010	50.70% ± 0.84	0.267 ± 0.012
ConSeq blosum62	54.32% ± 0.43	0.138 ± 0.016	63.81% ± 0.8	0.318 ± 0.014

[a] Data sets: The *PMD4k* data set contained 4 k proteins from the PMD (Kawabata et al. 1999); 74% of the SAVs were deemed effect in a binary classification. *DMS4* marks the first four human proteins (BRAC1, PTEN, TPMT, PPARG) for which we obtained comprehensive experimental DMS measurements along with a means of converting experimental scores into a binary version (effect/neutral) using synonyms. DMS4 results are shown for a threshold of 95%: the continuous effect scores were binarized by assigning the middle 95% of effect scores as neutral variants and SAVs resulting in effect scores outside this range as effect variants (Reeb et al. 2020). Methods: *SNAP2bin*: effect SNAP2 score > − 0.05, otherwise neutral; *VESPA*: effect score ≥ 0.5, otherwise neutral; *VESPAl*: effect score ≥ 0.5, otherwise neutral; *BLOSUM62*: negative BLOSUM62 scores predicted as effect, others as neutral; *ProtT5cons|ConSeq-19equal*: all 19 non-native SAVs predicted equally: effect if ProtT5cons|ConSeq predicted/labeled as conserved, otherwise neutral; *ProtT5beff|ConSeq-blosum62*: effect if ProtT5cons|ConSeq predicted/labeled as conserved and BLOSUM62 negative, otherwise neutral. ± values mark the 95% confidence interval (Eq. 12). For each column, if available, significantly best results are highlighted in bold

information in a more advanced way than standard profiles from MSAs appears to reach a similar top level without machine learning, namely GEMME (Laine et al. 2019), and so does a method based on probabilities from pLMs, namely ESM-1v, without using MSAs. Comparing all those to VESPA, we could not observe a single method outperforming all others on all DMS39 experiments (Fig. 5). The four methods compared (two using MSAs: DeepSequence and GEMME, two using probabilities from pLMs instead of MSAs: ESM-1v and VESPA) reached Spearman rank correlations above 0.4 for 36 DMS experiments. In fact, for the 11 highest correlating out of the 39 experiments, predictions were as accurate as typically the agreement between two different experimental studies of the same protein (Spearman 0.65 (Reeb et al. 2020)).

GEMME had a slightly higher mean and median Spearman correlation (Eq. 11) than DeepSequence, ESM-1v, VESPA, and all others tested (Fig. 6A, Table 2). When considering the symmetric 95% confidence intervals (Eq. 12), almost all those differences were statistically insignificant (Fig. 6B) except for only using BLOSUM62. In terms of mean Spearman correlation, VESPA was slightly higher than DeepSequence, which was slightly higher than ESM-1v (Fig. 6A), but again neither was significantly better. The median Spearman correlation was equal for ESM-1v and VESPA and insignificantly lower for DeepSequence. The fastest method, VESPAl, reached lower Spearman correlations than all other major methods (Fig. 6). Ranking and relative performance after correcting for statistical significance were identical for Spearman and Pearson correlation (Table S6).

For comparison, we also introduced two advances on a random baseline, namely the raw BLOSUM62 scores and the raw ProtT5 log-odds scores (Fig. 6; Fig. S7). BLOSUM62 was consistently and statistically significantly outperformed by all methods, while the ProtT5 log-odds averages were consistently lower, albeit not with statistical significance. As pLM-based methods were independent of MSAs, they predicted SAV scores for all residues contained in the DMS39 data sets, while, e.g., DeepSequence and GEMME could predict only for the subset of the residues covered by large enough MSAs. This was reflected by decreased coverage of methods relying on MSAs (DeepSequence and GEMME; Table S8). The Spearman correlation of ESM-1v, VESPA, and VESPAl for the SAVs in regions without MSAs was significantly lower than that in regions with MSAs available (Table S7).

SAV effect predictions blazingly fast: One important advantage of predicting SAV effects without using MSAs is the computational efficiency. For instance, to predict the

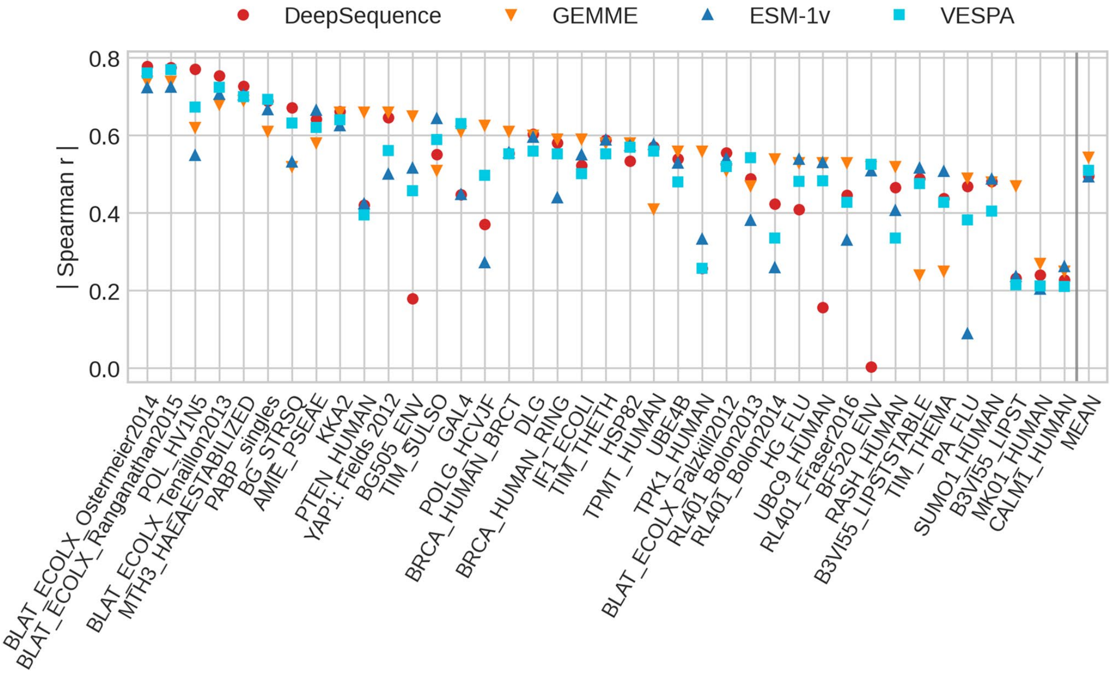

Fig. 5 No SAV effect prediction consistently best on DMS data. Data: *DMS39* (39 DMS experiments gathered for the development of DeepSequence (Riesselman et al. 2018)); experiments sorted by the maximum absolute Spearman coefficient for each experiment. Methods: **a** *DeepSequence* trained an unsupervised model for each DMS experiment using only MSA input, i.e., no effect score labels were used (Riesselman et al. 2018); **b** *GEMME* inferred evolutionary trees and conserved sites from MSAs to predict effects (Laine et al. 2019); **c** *ESM-1v* correlated log-odds of substitution probabilities (Methods) with SAV effect magnitudes (Meier et al. 2021); **d** *VESPA* (this work) trained a logistic regression ensemble on binary SAV classification (effect/neutral) using predicted conservation (ProtT5cons), BLOSUM62 (Henikoff and Henikoff 1992), and log-odds of substitution probabilities from ProtT5 (Elnaggar et al. 2021) as input (without any optimization on DMS data). The values for the absolute Spearman correlation (Eq. 11) are shown for each method and experiment. The rightmost column shows the mean absolute Spearman correlation for each method. Although some experiments correlated much better (toward left) with predictions than others (toward right), the spread between prediction methods appeared high for both extremes; DeepSequence was the only method reaching a correlation of 0 for one experiment; another one and three experiments were predicted with correlations below 0.2 for ESM-1v and DeepSequence, respectively, while the vast number of the 4×39 predictions reached correlations above 0.4

mutational effects for all 19 non-native SAVs in the entire human proteome (all residues in all human proteins) took 40 min on one Nvidia Quadro RTX 8000 using VESPAl. In turn, this was 40 min more than using BLOSUM62 alone (nearly instantaneous), but this instantaneous BLOSUM62-based prediction was also much worse (Q2 for binary BLOSUM62 prediction worse than random, Table 1). In contrast, running methods such as SNAP2 (or ConSeq) required first to generate MSAs. Even the blazingly fast MMseqs2 (Steinegger and Söding 2017) needed about 90 min using batch-processing on an Intel Skylake Gold 6248 processor with 40 threads, SSD and 377 GB main memory. While VESPAl computed prediction scores within minutes for an entire proteome, VESPA and ESM-1v require minutes for some single proteins depending on sequence length, e.g., ESM-1v took on average 170 s per protein for the DMS39 set, while ProtT5 required on average 780 s. This originated from the number of forward passes required to derive predictions: while VESPAl needed only a single forward pass through the pLM to derive embeddings for conservation prediction, VESPA and ESM-1v (when deriving "masked-marginals" as recommended by the authors) required L forward passes with L being the protein length, because they corrupt one amino acid at a time and try to reconstruct it. The large difference in runtime between ESM-1v and ProtT5 originated from the fact that ESM-1v cropped sequences after 1022, reducing the strong impact of outliers, i.e., runtime of transformer-based models scales quadratically with

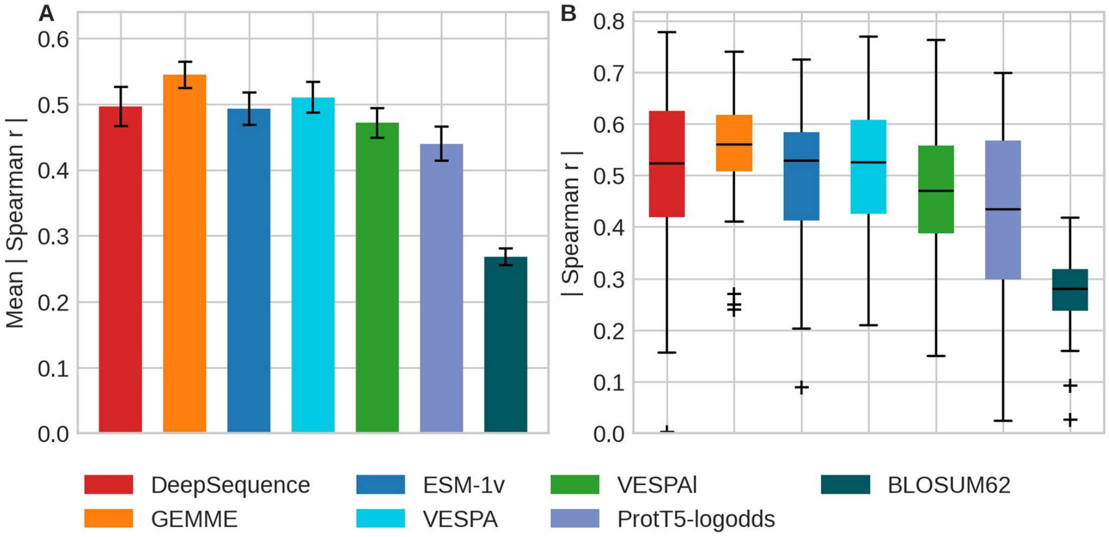

Fig. 6 Spearman correlation between prediction and DMS experiment varied. Data and methods as for Fig. 5 with addition of: *VESPAl*: fast version of VESPA with input limited to ProtT5cons and BLOSUM62; *ProtT5-logodds*: raw log-odds from ProtT5 embeddings (Elnaggar et al. 2021); and raw *BLOSUM62* substitution scores (Henikoff and Henikoff 1992). Panel A: mean absolute Spearman correlation coefficient (Eq. 11) for each method over all 39 DMS experiments; error bars highlight 0.95 confidence interval (1.96 standard errors). Ignoring statistical significance, the numerical ranking would be: GEMME, VESPA, DeepSequence, ESM-1v, VESPAl, ProtT5-logodds, and BLOSUM62. However, the first four did not differ by any statistical significance, and while those ranked 5 and 6 differed from the best four, 5 was close to 4, and 6 close to 5; only BLOSUM62, the raw substitution scores compiled as background were clearly worst. Panel B: boxplots on absolute Spearman correlation coefficients (Eq. 11) for each method over the 39 DMS experiments. The medians are depicted as black horizontal bars; whiskers are drawn at the 1.5 interquartile range

sequence length, so while the shortest protein (71 residues) in the DMS39 set took only 5 s to compute, the longest (3033 residues) took 4.5 h to compute. We leave investigating the effect of splitting very long proteins into (overlapping) chunks to future work.

Discussion

Conservation predicted by embeddings without MSAs. Even a simple logistic regression (LR) sufficed to predict per-residue conservation values from raw embeddings without using MSAs (Fig. 3, Table S1). Relatively shallow CNNs (with almost 100-times fewer free parameters than samples despite early stopping) improved over the LR to levels in predicting conservation only slightly below conservation assigned by ConSeq which explicitly uses MSAs (Fig. 3, Table S1). Did this imply that the pLMs extracted evolutionary information from unlabeled sequence databases (BFD (Steinegger and Söding 2018) and UniProt (The UniProt Consortium 2021))? The answer might be more elusive than it seems. The methodology (pLMs) applied to predict conservation never encountered any explicit information about protein families through MSAs, i.e., the pLMs used here never had an explicit opportunity to pick up evolutionary constraints from related proteins. The correlation between substitution probabilities derived from pLMs and conservation (Fig. 2) might suggest that pLMs implicitly learned evolutionary information.

A possible counterargument builds around the likelihood to pick up evolutionary constraints. The pLM clearly learned the reconstruction of more frequent amino acids much better than that of less frequent ones (Fig. S5). Unsurprisingly, AI is pushed most in the direction of most data. In fact, the differences between amino acid compositions were relatively small (less than factor of 10), suggesting that even an event occurring at one-tenth of the time may challenge pLMs. If the same pLM has to learn the evolutionary relation between two proteins belonging to the same family, it has to effectively master an event happening once in a million (assuming an average family size of about 2.5 k—thousand—in a database with 2.5b—billion—sequences). How can the model trip over a factor of 10^1 and at the same time master a factor of 10^6? Indeed, it seems almost impossible. If so, the pLM may not have learned evolutionary constraints, but the type of *bio-physical* constraint that also constrain evolution. In this interpretation, the pLM did not learn evolution, but

Table 2 Spearman correlation between SAV effect prediction and DMS experiments[a]

Method	Mean absolute r_S (Eq. 11)	Median absolute r_S (Eq. 11)
MSA-based		
DeepSequence	0.50 ± 0.03	0.52 ± 0.03
GEMME	0.53 ± 0.02	0.56 ± 0.02
pLM-based		
ESM-1v	0.49 ± 0.02	0.53 ± 0.02
VESPA	0.51 ± 0.02	0.53 ± 0.02
VESPAl	0.47 ± 0.02	0.47 ± 0.02

[a] Data sets: *DMS39* [39 DMS experiments gathered for the development of DeepSequence (Riesselman et al. 2018)] with 135,665 SAV scores. Methods: *DeepSequence*: AI trained on MSA for each of the DMS experiments (Riesselman et al. 2018); *GEMME*: using evolutionary information calculated from MSAs with few parameters optimized on DMS (Laine et al. 2019); *ESM-1v*: embedding-based prediction methods (Meier et al. 2021); *VESPA*: method developed here using logistic regression to combine predicted conservation (ProtT5cons), BLOSUM62 (Henikoff and Henikoff 1992) substitution scores, and log-odds from ProtT5 (Elnaggar et al. 2021); *VESPAl*: "light" version of VESPA using only predicted conservation and BLOSUM62 as input. ± values mark the standard error

the constraints "written into protein sequences" that determine which residue positions are more constrained.

In fact, one pLM used here, namely ProtT5, has recently been shown to explicitly capture aspects of long-range inter-residue distances directly during pre-training, i.e., without ever being trained on any labeled data pLMs pick up structural constraints that allow protein 3D structure prediction from single protein sequences (Weißenow et al. 2021). Another explanation for how ProtT5 embeddings capture conservation might be that pLMs picked up signals from short, frequently re-occurring sequence/structure motifs such as localization signals or catalytic sites that are more conserved than other parts of the sequence. If so, the pLM would not have to learn relationship between proteins but only between fragments, thereof reducing the factor 10^6 substantially. We could conceive of these motifs resembling some evolutionary nuclei, i.e., fragments shorter than structural domains that drove evolution (Alva et al. 2015; Ben-Tal and Lupas 2021; Kolodny 2021). Clearly, more work will have to shed light on the efficiency of (p)LMs in general (Bommasani et al. 2021).

Transformer-based pLMs best? We have tested a limited set of pLMs, largely chosen, because those had appeared to perform better than many other methods for a variety of different prediction tasks. Does the fact that in our hands Transformer-based pLMs worked best to predict residue conservation and SAVs imply that those will generally outperform other model types? By no means. While we expect that the about twenty approaches that we have compared in several of our recent methods (including the following 13: ESM-1[b|v] Meier et al. 2021; Rives et al. 2021), ProSE[*|DLM|MT] (Bepler and Berger 2019b, 2021), Prot[Albert|Bert|Electra|Vec|T5|T5XL|T5XLNet|T5XXL] (Elnaggar et al. 2021; Heinzinger et al. 2019) provided a somehow representative sampling of the existing options, our conclusions were only valid for embeddings extracted in a generic way from generic pLMs without any bearing on the methods underlying those pLMs.

Predicted conservation informative about SAV effects: DMS data sets with comprehensive experimental probing of the mutability landscape (Hecht et al. 2013) as, e.g., collected by MaveDB (Esposito et al. 2019) continue to pose problems for analysis, possibly due to a diversity of assays and protocols (Livesey and Marsh 2020; Reeb et al. 2020). Nevertheless, many such data sets capture important aspects about the susceptibility to change, i.e., the mutability landscape (Hecht et al. 2013). As always, the more carefully selected data sets become, the more they are used for the development of methods and therefore no longer can serve as independent data for assessments (Grimm et al. 2015; Reeb et al. 2016). Avoiding the traps of circularity and over-fitting by skipping training, our non-parametric rule-based approaches (ProtT5cons and ProtT5beff) suggested that predictions of SAV effects (by simply assigning "effect" to those SAVs where ProtT5cons predicted conserved and the corresponding BLOSUM62 value was negative) outperformed ConSeq with MSAs using the same idea, and even the expert effect prediction method SNAP2 (Fig. 4, Table 1).

Strictly speaking, it might be argued that one single free parameter was optimized using the data set, because for the PMD4k data set, the version that predicted the same effect for all 19-SAVs appeared to outperform the SAV-specific prediction using BLOSUM62 (*19equal* vs *blosum62* in Fig. 4 and Table 1). However, not even the values computed for PMD4k could distract from the simple fact that not all SAVs are equal, i.e., that regardless of model performance, *19equal* will not be used exclusively for any method. In fact, the concept of combining predictions with BLOSUM62 values has been shown to succeed for function prediction before (Bromberg and Rost 2008; Schelling et al. 2018) in that sense it was arguably not an optimizable hyperparameter. Embeddings predicted conservation (Fig. 3); conservation predicted SAV effects (Fig. 4). Did this imply that embeddings captured evolutionary information? Once again, we could not answer this question either way directly. To repeat: our procedure/method never used information from MSAs in any way. Could it have implicitly learned this? To repeat the previous speculation: embeddings *might* capture a reality that constrains what can be observed in evolution, and this reality is exactly what is used for the part of the SAV effect prediction that succeeds. If so, we would argue that our simplified method did not succeed, because it predicted conservation without using MSAs, but that it captured positions biophysically "marked by constraints", i.e., residues with higher contact density in protein 3D structures (Weißenow et al. 2021). This assumption would explain how predicted conservation (ProtT5cons) not using evolutionary information could predict SAV effects better than a slightly more correct approach (ConSeq) using MSAs to extract evolutionary information (Fig. 4: ProtT5cons vs. ConSeq).

Substitution probabilities from pLMs capture aspects measured by DMS experiments: Using embeddings to predict SAV effects through conservation prediction succeeded but appeared like a detour. ESM-1v (Meier et al. 2021) pioneered a direct path from reconstruction/substitution probabilities of pLMs to SAV effect predictions. When comparing the ESM-1v encoder-based with the ProtT5 encoder–decoder-based Transformer, we encountered surprising results. Previously, ProtT5 usually performed at least on par with previous versions of ESM (e.g., ESM-1b (Rives et al. 2021)) or outperformed them (Elnaggar et al. 2021). In contrast, the substitution probabilities of ProtT5 were clearly inferior to those from ESM-1v in their correlation with the 39 DMS experiments (Fig. 6). This reversed trend might have resulted from a combination of the following facts: (1) ProtT5 is a single model, while ESM-1v is an ensemble of five pLMs potentially leading to a smoother substitution score. (2) ESM-1v was trained on UniRef90 instead of BFD/UniRef50 (ProtT5) possibly providing a broader view on the mutability landscape of proteins. In fact, the ESM-1v authors showed a significant improvement when pre-training on UniRef90 instead of UniRef50 (Rives et al. 2021). (3) ESM-1v is a BERT-style, encoder-based Transformer, while ProtT5 is based on T5's encoder-decoder structure. In previous experiments (Elnaggar et al. 2021), we only extracted embeddings from ProtT5's encoder (e.g., ProtT5cons is based on encoder embeddings), because its decoder fell significantly short in all experiments. However, only T5's decoder can output probabilities, so we had to fall back to ProtT5's decoder for SAV effect predictions. This discrepancy of encoder and decoder performance can only be sketched here. In short, encoder-based transformer models always *see* the context of the whole sequence (as does ProtT5 's encoder and ESM-1v), while decoder-based transformer models (such as ProtT5's decoder or GPT (Radford et al. 2019)) *see* only single-sided context, because they are generating text (sequence-to-sequence models (Sutskever et al. 2014)). This is crucial for translation tasks, but appeared sub-optimal in our setting. Despite this shortcoming in performance, we trained VESPA based on log-odds derived from ProtT5 substitution probabilities, mainly because we started this work before the release of ESM-1v. Also, we hoped for synergy effects when implementing VESPA into the PredictProtein webserver, because ProtT5 is already used by many of our predictors. Finding the best combination of pLM substitution probabilities for SAV effect prediction will remain subject for future work.

Fast predictions save computing resources? Our simple protocol introduced here enabled extremely efficient, speedy predictions. While pre-training pLMs consumed immense resources (Elnaggar et al. 2021), this was done in the past. The new development here was the models for the 2nd level supervised transfer learning. Inputting ProtT5 embeddings to predict residue conservation (ProtT5cons) or SAV effects (VESPA/VESPAl) for predictions in the future will consume very little additional resources. When running prediction servers such as PredictProtein (Bernhofer et al. 2021) queried over 3000 times every month, such investments could be recovered rapidly at seemingly small prices to pay even if performance was slightly reduced. How to quantify this? At what gain in computing efficiency is which performance reduction acceptable? Clearly, there will not be one answer for all purposes, but the recent reports on climate change strongly suggest to begin considering such questions.

Quantitative metrics for hypothetical improvements over MSA-based methods? If methods using single sequences without MSAs perform as well as, or even better than, SOTA methods using MSAs, could we quantify metrics measuring the hypothetical improvements from embeddings? This question raised by an anonymous reviewer opens an interesting new perspective. Gain in speed, reduction of computational costs clearly could evolve as one such metric. A related issue is related to protein design: for some applications, the difference in speed might open new doors. Although we have no data to show for others, we could imagine yet another set of metrics measuring the degree to which embedding-based methods realize more protein-specific than family averaged predictions.

Conclusions

Embeddings extracted from protein Language Models (pLMs, Fig. 1), namely from ProtBert and ProtT5 (Elnaggar et al. 2021) and ESM-1b (Rives et al. 2021), contain information that sufficed to predict residue conservation in protein families without using multiple sequence alignments (MSAs, Fig. 3). Such predictions of conservation combined with BLOSUM62 scores predicted the effects of sequence variation (single amino acid variants, or SAVs) without optimizing any additional free parameter (*ProtT5beff*, Fig. 6). Through further training on binary experimental data (effect/neutral), we developed *VESPA*, a relatively simple, yet apparently successful new method for SAV effect prediction (Fig. 4). This method even worked so well on non-binary data from 39 DMS experiments that without ever using such data nor ever using MSAs; VESPA appeared competitive with the SOTA (Fig. 5, Fig. 6), although for SAV effect predictions, embedding-based methods are still not yet outperforming the MSA-based SOTA as for other prediction tasks (Elnaggar et al. 2021; Littmann et al. 2021a, b, c; Stärk et al. 2021). Embedding-based predictions are blazingly fast, thereby they save computing, and ultimately energy resources when applied to daily sequence analysis. In combination, our results suggested that the major signal captured by variant effect predictions originates from some biophysical constraint revealed by raw protein sequences. The ConSurf10k dataset is available at https://doi.org/10.5281/zenodo.5238537. For high-throughput predictions, methods are available through bio_embeddings (Dallago et al. 2021). For single queries VESPA and ProtT5cons will be made available through the PredictProtein server (Bernhofer et al. 2021). VESPA and VESPAl are also available from github at https://github.com/Rostlab/VESPA.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00439-021-02411-y.

Acknowledgements Thanks to Tim Karl and Inga Weise (both TUM) for invaluable help with technical and administrative aspects of this work. Thanks to Nir Ben-Tal (Tel Aviv U) and his team for the excellent services around ConSurf-DB and ConSeq; to Yana Bromberg (Rutgers U), Max Hecht (Amazon) for advancing SNAP; to Adam Riesselman, John Ingraham, and Debbie Marks (Harvard) for making their collection of DMS data available; to Elodie Laine (Sorbonne U) for providing the predictions of GEMME; to the group around Facebook AI Research for making ESM-1b and ESM-1v readily available; to the Dunbrack lab for Pisces, and most importantly to Martin Steinegger (Seoul Natl. Univ.) and his team for MMseqs2 and BFD. Particular thanks to two anonymous reviewers who helped crucially with improving this work and to the valuable comments from the editors. Last, but not least, thanks to all who deposit their experimental data in public databases, and to those who maintain these databases.

Author contributions CM implemented and evaluated the methods and took the lead in writing the manuscript. MH conceived, trained, and evaluated the neural networks on conservation prediction, contributed ideas, and proofread the manuscript. TO and CD contributed crucial ideas and provided valuable comments. MB helped in generating the evaluation methods ConSeq and SNAP2. KE supported the work with coding advice and created the original ConSurf10k data set. DN contributed the clusters and subsets of the SNAP2 development set. BR supervised and guided the work and co-wrote the manuscript. All authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the DFG grant RO 1320/4-1, Software Campus Funding (BMBF 01IS17049) and the KONWIHR Program and by the Bavarian Ministry for Education through funding to the TUM.

Declarations

Conflict of interest No author declares any competing interest.

References

Adzhubei IA, Schmidt S, Peshkin L, Ramensky VE, Gerasimova A, Bork P, Kondrashov AS, Sunyaev SR (2010) A method and server for predicting damaging missense mutations. Nat Methods 7:248–249. https://doi.org/10.1038/nmeth0410-248

Alley EC, Khimulya G, Biswas S, AlQuraishi M, Church GM (2019) Unified rational protein engineering with sequence-based deep representation learning. Nat Methods 16:1315–1322. https://doi.org/10.1038/s41592-019-0598-1

Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25:3389–3402. https://doi.org/10.1093/nar/25.17.3389

Alva V, Söding J, Lupas AN (2015) A vocabulary of ancient peptides at the origin of folded proteins. Elife. https://doi.org/10.7554/eLife.09410

Amberger JS, Bocchini CA, Scott AF, Hamosh A (2019) OMIM.org: leveraging knowledge across phenotype-gene relationships. Nucleic Acids Res 47:D1038–D1043. https://doi.org/10.1093/nar/gky1151

AVE Alliance Founding Members (2020) Atlas of Variant Effect Alliance.

Ben Chorin A, Masrati G, Kessel A, Narunsky A, Sprinzak J, Lahav S, Ashkenazy H, Ben-Tal N (2020) ConSurf-DB: An accessible repository for the evolutionary conservation patterns of the majority of PDB proteins. Protein Sci 29:258–267. https://doi.org/10.1002/pro.3779

Ben-Tal N, Lupas AN (2021) Editorial overview: Sequences and topology: 'paths from sequence to structure.' Curr Opin Struct Biol. https://doi.org/10.1016/j.sbi.2021.05.005

Bepler T, Berger B (2019a) Learning protein sequence embeddings using information from structure. arXiv. https://arxiv.org/abs/astro-ph/1902.08661

Bepler T, Berger B (2019b) Learning protein sequence embeddings using information from structure Seventh International Conference on Learning Representations

Bepler T, Berger B (2021) Learning the protein language: evolution, structure, and function. Cell Syst 12(654–669):e3. https://doi.org/10.1016/j.cels.2021.05.017

Berezin C, Glaser F, Rosenberg J, Paz I, Pupko T, Fariselli P, Casadio R, Ben-Tal N (2004) ConSeq: the identification of functionally and structurally important residues in protein sequences. Bioinformatics (oxford, England) 20:1322–1324. https://doi.org/10.1093/bioinformatics/bth070

Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. Nucleic Acids Res 28:235–242. https://doi.org/10.1093/nar/28.1.235

Bernhofer M, Dallago C, Karl T, Satagopam V, Heinzinger M, Littmann M, Olenyi T, Qiu J, Schutze K, Yachdav G, Ashkenazy H, Ben-Tal N, Bromberg Y, Goldberg T, Kajan L, O'Donoghue S, Sander C, Schafferhans A, Schlessinger A, Vriend G, Mirdita M, Gawron P, Gu W, Jarosz Y, Trefois C, Steinegger M, Schneider R, Rost B (2021) PredictProtein—predicting protein structure and function for 29 years. Nucleic Acids Res. https://doi.org/10.1093/nar/gkab354

Bommasani R, Hudson DA, Adeli E, Altman R, Arora S, von Arx S, Bernstein MS, Bohg J, Bosselut A, Brunskill E, Brynjolfsson E, Buch S, Card D, Castellon R, Chatterji N, Chen A, Creel K, Quincy Davis J, Demszky D, Donahue C, Doumbouya M, Durmus E, Ermon S, Etchemendy J, Ethayarajh K, Fei-Fei L, Finn C, Gale T, Gillespie L, Goel K, Goodman N, Grossman S, Guha N, Hashimoto T, Henderson P, Hewitt J, Ho DE, Hong J, Hsu K, Huang J, Icard T, Jain S, Jurafsky D, Kalluri P, Karamcheti S, Keeling G, Khani F, Khattab O, Kohd PW, Krass M, Krishna R, Kuditipudi R, Kumar A, Ladhak F, Lee M, Lee T, Leskovec J, Levent I, Li XL, Li X, Ma T, Malik A, Manning CD, Mirchandani S, Mitchell E, Munyikwa Z, Nair S, Narayan A, Narayanan D, Newman B, Nie A, Niebles JC, Nilforoshan H, Nyarko J, Ogut G, Orr L, Papadimitriou I, Park JS, Piech C, Portelance E, Potts C, Raghunathan A, Reich R, Ren H, Rong F, Roohani Y, Ruiz C, Ryan J, Ré C, Sadigh D, Sagawa S, Santhanam K, Shih A, Srinivasan K, Tamkin A, Taori R, Thomas AW, Tramèr F, Wang RE, Wang W, et al. (2021) On the Opportunities and Risks of Foundation Models. https://arxiv.org/abs/astro-ph/2108.07258

Bromberg Y, Rost B (2007) SNAP: predict effect of non-synonymous polymorphisms on function. Nucleic Acids Res 35:3823–3835

Bromberg Y, Rost B (2008) Comprehensive in silico mutagenesis highlights functionally important residues in proteins. Bioinformatics 24:i207–i212

Bromberg Y, Rost B (2009) Correlating protein function and stability through the analysis of single amino acid substitutions. BMC Bioinformatics 10:S8. https://doi.org/10.1186/1471-2105-10-s8-s8

Burley SK, Berman HM, Bhikadiya C, Bi C, Chen L, Di Costanzo L, Christie C, Dalenberg K, Duarte JM, Dutta S, Feng Z, Ghosh S, Goodsell DS, Green RK, Guranovic V, Guzenko D, Hudson BP, Kalro T, Liang Y, Lowe R, Namkoong H, Peisach E, Periskova I, Prlic A, Randle C, Rose A, Rose P, Sala R, Sekharan M, Shao C, Tan L, Tao YP, Valasatava Y, Voigt M, Westbrook J, Woo J, Yang H, Young J, Zhuravleva M, Zardecki C (2019) RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. Nucleic Acids Res 47:D464–D474. https://doi.org/10.1093/nar/gky1004

Capriotti E, Fariselli P, Casadio R (2005) I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. Nucleic Acids Res 33:W306–W310. https://doi.org/10.1093/nar/gki375

Dallago C, Schuetze K, Heinzinger M, Olenyi T, Littmann M, Lu AX, Yang KK, Min S, Yoon S, Morton JT, Rost B (2021) Learned embeddings from deep learning to visualize and predict protein sets. Curr Protoc 1:e113. https://doi.org/10.1002/cpz1.113

Devlin J, Chang M-W, Lee K, Toutanova K (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/astro-ph/1810.04805 [cs]

Efron B, Halloran E, Holmes S (1996) Bootstrap confidence levels for phylogenetic trees. Proc Nat Acad Sci USA 93:13429–13434

Elnaggar A, Heinzinger M, Dallago C, Rehawi G, Wang Y, Jones L, Gibbs T, Feher T, Angerer C, Steinegger M, Bhowmik D, Rost B (2021) ProtTrans: towards cracking the language of life's code through self-supervised learning. Mach Intell 14:30

Esposito D, Weile J, Shendure J, Starita LM, Papenfuss AT, Roth FP, Fowler DM, Rubin AF (2019) MaveDB: an open-source platform to distribute and interpret data from multiplexed assays of variant effect. Genome Biol 20:223. https://doi.org/10.1186/s13059-019-1845-6

Fariselli P, Martelli PL, Savojardo C, Casadio R (2015) INPS: predicting the impact of non-synonymous variations on protein stability from sequence. Bioinformatics 31:2816–2821. https://doi.org/10.1093/bioinformatics/btv291

Findlay GM, Daza RM, Martin B, Zhang MD, Leith AP, Gasperini M, Janizek JD, Huang X, Starita LM, Shendure J (2018) Accurate classification of BRCA1 variants with saturation genome editing. Nature 562:217–222. https://doi.org/10.1038/s41586-018-0461-z

Fowler DM, Fields S (2014) Deep mutational scanning: a new style of protein science. Nat Methods 11:801–807. https://doi.org/10.1038/nmeth.3027

Fu L, Niu B, Zhu Z, Wu S, Li W (2012) CD-HIT: accelerated for clustering the next-generation sequencing data. Bioinformatics 28:3150–3152. https://doi.org/10.1093/bioinformatics/bts565

Fukushima K (1969) Visual feature extraction by a multilayered network of analog threshold elements. IEEE Trans Syst Sci Cybern 5:322–333. https://doi.org/10.1109/TSSC.1969.300225

Gray VE, Hause RJ, Luebeck J, Shendure J, Fowler DM (2018) Quantitative missense variant effect prediction using large-scale mutagenesis data. Cell Syst 6:116-124.e3. https://doi.org/10.1016/j.cels.2017.11.003

Grimm DG, Azencott CA, Aicheler F, Gieraths U, Macarthur DG, Samocha KE, Cooper DN, Stenson PD, Daly MJ, Smoller JW, Duncan LE, Borgwardt KM (2015) The evaluation of tools used to predict the impact of missense variants is hindered by two types of circularity. Hum Mutat 36:513–523. https://doi.org/10.1002/humu.22768

Hecht M, Bromberg Y, Rost B (2013) News from the protein mutability landscape. J Mol Biol 425:3937–3948. https://doi.org/10.1016/j.jmb.2013.07.028

Hecht M, Bromberg Y, Rost B (2015) Better prediction of functional effects for sequence variants. BMC Genomics 16:S1. https://doi.org/10.1186/1471-2164-16-s8-s1

Heinzinger M, Elnaggar A, Wang Y, Dallago C, Nechaev D, Matthes F, Rost B (2019) Modeling aspects of the language of life through transfer-learning protein sequences. BMC Bioinformatics 20:723. https://doi.org/10.1186/s12859-019-3220-8

Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci 89:10915–10919. https://doi.org/10.1073/pnas.89.22.10915

Hopf TA, Ingraham JB, Poelwijk FJ, Scharfe CP, Springer M, Sander C, Marks DS (2017) Mutation effects predicted from sequence co-variation. Nat Biotechnol 35:128–135. https://doi.org/10.1038/nbt.3769

Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, Tunyasuvunakool K, Bates R, Zidek A, Potapenko A, Bridgland A, Meyer C, Kohl SAA, Ballard AJ, Cowie A, Romera-Paredes B, Nikolov S, Jain R, Adler J, Back T, Petersen S, Reiman D, Clancy E, Zielinski M, Steinegger M, Pacholska M, Berghammer T, Bodenstein S, Silver D, Vinyals O, Senior AW, Kavukcuoglu K, Kohli P, Hassabis D (2021) Highly accurate protein structure prediction with AlphaFold. Nature. https://doi.org/10.1038/s41586-021-03819-2

Katoh K, Standley DM (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30:772–780. https://doi.org/10.1093/molbev/mst010

Katsonis P, Lichtarge O (2014) A formal perturbation equation between genotype and phenotype determines the Evolutionary Action of protein-coding variations on fitness. Genome Res 24:2050–2058. https://doi.org/10.1101/gr.176214.114

Kawabata T, Ota M, Nishikawa K (1999) The protein mutant database. Nucleic Acids Res 27:355–357. https://doi.org/10.1093/nar/27.1.355

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. pp https://arxiv.org/abs/astro-ph/1412.6980

Kircher M, Witten DM, Jain P, O'Roak BJ, Cooper GM, Shendure J (2014) A general framework for estimating the relative pathogenicity of human genetic variants. Nat Genet 46:310–315. https://doi.org/10.1038/ng.2892

Kolodny R (2021) Searching protein space for ancient sub-domain segments. Curr Opin Struct Biol 68:105–112. https://doi.org/10.1016/j.sbi.2020.11.006

Korber B, Fischer WM, Gnanakaran S, Yoon H, Theiler J, Abfalterer W, Hengartner N, Giorgi EE, Bhattacharya T, Foley B, Hastie KM, Parker MD, Partridge DG, Evans CM, Freeman TM, de Silva TI, Angyal A, Brown RL, Carrilero L, Green LR, Groves DC, Johnson KJ, Keeley AJ, Lindsey BB, Parsons PJ, Raza M, Rowland-Jones S, Smith N, Tucker RM, Wang D, Wyles MD, McDanal C, Perez LG, Tang H, Moon-Walker A, Whelan SP, LaBranche CC, Saphire EO, Montefiori DC (2020) Tracking changes in SARS-CoV-2 spike: evidence that D614G increases infectivity of the COVID-19 virus. Cell 182:812-827.e19. https://doi.org/10.1016/j.cell.2020.06.043

Laha S, Chakraborty J, Das S, Manna SK, Biswas S, Chatterjee R (2020) Characterizations of SARS-CoV-2 mutational profile, spike protein stability and viral transmission. Infect Genet Evol 85:104445. https://doi.org/10.1016/j.meegid.2020.104445

Laine E, Karami Y, Carbone A (2019) GEMME: a simple and fast global epistatic model predicting mutational effects. Mol Biol Evol. https://doi.org/10.1093/molbev/msz179

Littmann M, Bordin N, Heinzinger M, Schütze K, Dallago C, Orengo C, Rost B (2021a) Clustering funFams using sequence embeddings improves EC purity. Bioinformatics. https://doi.org/10.1093/bioinformatics/btab371

Littmann M, Heinzinger M, Dallago C, Olenyi T, Rost B (2021b) Embeddings from deep learning transfer GO annotations beyond homology. Sci Rep 11:1160. https://doi.org/10.1038/s41598-020-80786-0

Littmann M, Heinzinger M, Dallago C, Weissenow K, Rost B (2021c) Protein embeddings and deep learning predict binding residues for various ligand classes. bioRxiv. https://doi.org/10.1101/2021.09.03.458869

Liu J, Rost B (2003) Domains, motifs, and clusters in the protein universe. Curr Opin Chem Biol 7:5–11

Liu J, Rost B (2004a) CHOP proteins into structural domain-like fragments. Proteins: structure. Funct Bioinf 55:678–688

Liu J, Rost B (2004b) Sequence-based prediction of protein domains. Nucleic Acids Res 32:3522–3530

Livesey BJ, Marsh JA (2020) Using deep mutational scanning to benchmark variant effect predictors and identify disease mutations. Mol Syst Biol 16:e9380. https://doi.org/10.15252/msb.20199380

Madani A, McCann B, Naik N, Shirish Keskar N, Anand N, Eguchi RR, Huang P, Socher R (2020) ProGen: language modeling for protein generation. arXiv 16:1315

Majithia AR, Tsuda B, Agostini M, Gnanapradeepan K, Rice R, Peloso G, Patel KA, Zhang X, Broekema MF, Patterson N, Duby M, Sharpe T, Kalkhoven E, Rosen ED, Barroso I, Ellard S, UKMD Consortium, Kathiresan S, Myocardial Infarction Genetics, O'Rahilly S, UKCL Consortiun, Chatterjee K, Florez JC, Mikkelsen T, Savage DB, Altshuler D (2016) Prospective functional classification of all possible missense variants in PPARG. Nat Genet 48:1570–1575. https://doi.org/10.1038/ng.3700

Matreyek KA, Starita LM, Stephany JJ, Martin B, Chiasson MA, Gray VE, Kircher M, Khechaduri A, Dines JN, Hause RJ, Bhatia S, Evans WE, Relling MV, Yang W, Shendure J, Fowler DM (2018) Multiplex assessment of protein variant abundance by massively parallel sequencing. Nat Genet 50:874–882. https://doi.org/10.1038/s41588-018-0122-z

Meier J, Rao R, Verkuil R, Liu J, Sercu T, Rives A (2021) Language models enable zero-shot prediction of the effects of mutations on protein function. bioRxiv. https://doi.org/10.1101/2021.07.09.450648

Mercatelli D, Giorgi FM (2020) Geographic and genomic distribution of SARS-CoV-2 mutations. Front Microbiol. https://doi.org/10.3389/fmicb.2020.01800

Miller M, Bromberg Y, Swint-Kruse L (2017) Computational predictors fail to identify amino acid substitution effects at rheostat positions. Sci Rep 7:41329. https://doi.org/10.1038/srep41329

Mistry J, Finn RD, Eddy SR, Bateman A, Punta M (2013) Challenges in homology search: HMMER3 and convergent evolution of coiled-coil regions. Nucleic Acids Res 41:e121. https://doi.org/10.1093/nar/gkt263

Ng PC, Henikoff S (2003) SIFT: predicting amino acid changes that affect protein function. Nucleic Acids Res 31:3812–3814

Niroula A, Urolagin S, Vihinen M (2015) PON-P2: prediction method for fast and reliable identification of harmful variants. PLoS ONE 10:e0117380. https://doi.org/10.1371/journal.pone.0117380

Nishikawa K, Ishino S, Takenaka H, Norioka N, Hirai T, Yao T, Seto Y (1994) Constructing a protein mutant database. Protein Eng 7:733. https://doi.org/10.1093/protein/7.5.733

O'Donoghue SI, Schafferhans A, Sikta N, Stolte C, Kaur S, Ho BK, Anderson S, Procter J, Dallago C, Bordin N, Adcock M, Rost B (2020) SARS-CoV-2 structural coverage map reveals state changes that disrupt host immunity. bioRxiv. https://doi.org/10.1101/2020.07.16.207308

Ofer D, Brandes N, Linial M (2021) The language of proteins: NLP, machine learning and protein sequences. Comput Struct Biotechnol J 19:1750–1758. https://doi.org/10.1016/j.csbj.2021.03.022

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830

Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019) Language Models are Unsupervised Multitask Learners. 24.

Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu PJ (2020) Exploring the limits of transfer learning with a unified text-to-text transformer. https://arxiv.org/abs/astro-ph/1910.10683[cs, stat].

Ramensky V, Bork P, Sunyaev S (2002) Human non-synonymous SNPs: server and survey. Nucleic Acids Res 30:3894–3900

Rao R, Meier J, Sercu T, Ovchinnikov S, Rives A (2020) Transformer protein language models are unsupervised structure learners. bioRxiv. https://doi.org/10.1101/2020.12.15.422761

Reeb J (2020) Data for: Variant effect predictions capture some aspects of deep mutational scanning experiments. 1. doi: https://doi.org/10.17632/2rwrkp7mfk.1

Reeb J, Hecht M, Mahlich Y, Bromberg Y, Rost B (2016) Predicted molecular effects of sequence variants link to system level of disease. PLoS Comput Biol 12:e1005047. https://doi.org/10.1371/journal.pcbi.1005047

Reeb J, Wirth T, Rost B (2020) Variant effect predictions capture some aspects of deep mutational scanning experiments. BMC Bioinf 21:107. https://doi.org/10.1186/s12859-020-3439-4

Riesselman AJ, Ingraham JB, Marks DS (2018) Deep generative models of genetic variation capture the effects of mutations. Nat Methods 15:816–822. https://doi.org/10.1038/s41592-018-0138-4

Rives A, Meier J, Sercu T, Goyal S, Lin Z, Liu J, Guo D, Ott M, Zitnick CL, Ma J, Fergus R (2021) Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Proc Natl Acad Sci. https://doi.org/10.1073/pnas.2016239118

Rost B (1996) PHD: predicting one-dimensional protein structure by profile based neural networks. Methods Enzymol 266:525–539

Rost B, Sander C (1992) Jury returns on structure prediction. Nature 360:540

Rost B, Sander C (1993) Prediction of protein secondary structure at better than 70% accuracy. J Mol Biol 232:584–599. https://doi.org/10.1006/jmbi.1993.1413

Schelling M, Hopf TA, Rost B (2018) Evolutionary couplings and sequence variation effect predict protein binding sites. Proteins 86:1064–1074. https://doi.org/10.1002/prot.25585

Schwarz JM, Rodelsperger C, Schuelke M, Seelow D (2010) MutationTaster evaluates disease-causing potential of sequence alterations. Nat Methods 7:575–576. https://doi.org/10.1038/nmeth0810-575

Sim N-L, Kumar P, Hu J, Henikoff S, Schneider G, Ng PC (2012) SIFT web server: predicting effects of amino acid substitutions on proteins. Nucleic Acids Res 40:W452–W457. https://doi.org/10.1093/nar/gks539

Sruthi CK, Balaram H, Prakash MK (2020) Toward developing intuitive rules for protein variant effect prediction using deep mutational scanning data. ACS Omega 5:29667–29677. https://doi.org/10.1021/acsomega.0c02402

Stärk H, Dallago C, Heinzinger M, Rost B (2021) Light attention predicts protein location from the language of life. bioRxiv. https://doi.org/10.1101/2021.04.25.441334

Steinegger M, Söding J (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nat Biotechnol 35:1026

Steinegger M, Söding J (2018) Clustering huge protein sequence sets in linear time. Nat Commun 9:2542. https://doi.org/10.1038/s41467-018-04964-5

Studer RA, Dessailly BH, Orengo CA (2013) Residue mutations and their impact on protein structure and function: detecting beneficial and pathogenic changes. Biochem J 449:581–594. https://doi.org/10.1042/BJ20121221

Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. Proceedings of the 27th International Conference on Neural Information Processing Systems—Volume 2. MIT Press, Montreal, Canada, pp 3104–3112

The UniProt Consortium (2021) UniProt: the universal protein knowledgebase in 2021. Nucleic Acids Res 49:D480–D489. https://doi.org/10.1093/nar/gkaa1100

Wang G, Dunbrack RL Jr (2003) PISCES: a protein sequence culling server. Bioinformatics 19:1589–1591. https://doi.org/10.1093/bioinformatics/btg224

Wang Z, Moult J (2001) SNPs, protein structure, and disease. Hum Mutat 17:263–270. https://doi.org/10.1002/humu.22

Weile J, Roth FP (2018) Multiplexed assays of variant effects contribute to a growing genotype–phenotype atlas. Hum Genet 137:665–678. https://doi.org/10.1007/s00439-018-1916-x

Weißenow K, Heinzinger M, Rost B (2021) Protein language model embeddings for fast, accurate, alignment-free protein structure prediction. bioRxiv. https://doi.org/10.1101/2021.07.31.454572

Zhou G, Chen M, Ju CJT, Wang Z, Jiang JY, Wang W (2020) Mutation effect estimation on protein-protein interactions using deep contextualized representation learning. NAR Genom Bioinform. https://doi.org/10.1093/nargab/lqaa015

5. Contrastive Learning on Protein Embeddings Enlightens Midnight Zone

5.1. Preface

Experimental 3D structures of proteins are leveraged through multiple sequence alignments (MSAs), or more generally through homology-based inference (HBI), facilitating the transfer of information from a protein with known annotation to a query without any annotation. A recent alternative expands the concept of HBI from sequence-distance lookup to embedding-based annotation transfer (Chapter 3 - EAT) by transferring annotations by means of Euclidean distance between vector representations (embeddings) derived from protein Language Models (pLMs; Chapter 2 - SeqVec and [28]).

Here, we showcased how contrastive learning can be used to improve over those embeddings by refining them for a specific task. This learning procedure creates a new set of embeddings that optimizes constraints captured by the hierarchical classifications of protein 3D structures defined by the CATH [17] resource. More specifically, we optimize the new embedding space towards pushing apart proteins with dissimilar CATH annotation while pulling together those proteins with similar annotations. Importantly, this is done simultaneously for all hierarchy-levels in CATH which allows to bypass the problems arising from traditional approaches, which force the CATH hierarchy into a classical machine learning classification problem with thousands of classes. The approach, dubbed ProtTucker, has an improved ability to recognize distant homologous relationships compared to more traditional techniques such as threading or fold recognition. Thus, these new embeddings have allowed sequence comparison to step into the “midnight zone” [56] of protein similarity, i.e., the region in which distantly related sequences have a seemingly random pairwise sequence similarity. The novelty of this work is in the particular combination of tools and sampling techniques that ascertained good performance comparable or better to existing state-of-the-art sequence comparison methods. Additionally, since this method does not need to generate alignments, it is also orders of magnitudes faster. The code and some pre-computed lookup databases such as CATH are available at https://github.com/Rostlab/EAT .

Author contribution: I am responsible for the original conceptualisation, performed the training and evaluation, created the GitHub repository and wrote the initial draft of the manuscript. Maria Littmann performed the evaluation on FunFams. Nicola Bordin provided additional data with entries shared by CATH and SCOP. All authors drafted the manuscript.

5.2. Journal Article: Heinzinger *et al.*, NAR Genomics and Bioinformatics (2022)

Reference: Heinzinger, M., Littmann, M., Sillitoe, I., Bordin, N., Orengo, C., et al. Contrastive learning on protein embeddings enlightens midnight zone. *NAR Genomics and Bioinformatics*, 4(2), 2022. ISSN 2631-9268. 10.1093/nargab/lqac043. Lqac043

https://doi.org/10.1093/nargab/lqac043

Contrastive learning on protein embeddings enlightens midnight zone

Michael Heinzinger[1,2,*], Maria Littmann[1], Ian Sillitoe[3], Nicola Bordin[3], Christine Orengo[3] and Burkhard Rost[1,4,*]

[1]TUM (Technical University of Munich) Dept Informatics, Bioinformatics & Computational Biology - i12, Boltzmannstr. 3, 85748 Garching/Munich, Germany, [2]TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany, [3]Institute of Structural and Molecular Biology, University College London, London WC1E 6BT, UK and [4]Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, 85748 Garching, Germany & TUM School of Life Sciences Weihenstephan (WZW), Alte Akademie 8, Freising, Germany

Received November 22, 2021; Revised March 25, 2022; Editorial Decision May 09, 2022; Accepted May 17, 2022

ABSTRACT

Experimental structures are leveraged through multiple sequence alignments, or more generally through homology-based inference (HBI), facilitating the transfer of information from a protein with known annotation to a query without any annotation. A recent alternative expands the concept of HBI from sequence-distance lookup to embedding-based annotation transfer (EAT). These embeddings are derived from protein Language Models (pLMs). Here, we introduce using single protein representations from pLMs for contrastive learning. This learning procedure creates a new set of embeddings that optimizes constraints captured by hierarchical classifications of protein 3D structures defined by the CATH resource. The approach, dubbed *ProtTucker*, has an improved ability to recognize distant homologous relationships than more traditional techniques such as threading or fold recognition. Thus, these embeddings have allowed sequence comparison to step into the 'midnight zone' of protein similarity, i.e. the region in which distantly related sequences have a seemingly random pairwise sequence similarity. The novelty of this work is in the particular combination of tools and sampling techniques that ascertained good performance comparable or better to existing state-of-the-art sequence comparison methods. Additionally, since this method does not need to generate alignments it is also orders of magnitudes faster. The code is available at https://github.com/Rostlab/EAT.

INTRODUCTION

Phase-transition from daylight through twilight into midnight zone

Protein sequence determines structure which determines function. This simple chain underlies the success of grouping proteins into families from sequence (1–4). Information from experimental high-resolution three-dimensional (3D) structures expands the perspective from families to superfamilies (5,6) that often reveal evolutionary and functional connections not recognizable from sequence alone (7,8). Thus, 3D information helps us to penetrate through the twilight zone of sequence alignments (9,10) into the midnight zone of distant evolutionary relationships (11).

The transition from daylight, through twilight and into the midnight zone is characterized by a phase-transition, i.e. a sigmoid function describing an order of magnitude increase in recall (relations identified) at the expense of a decrease in precision (relations identified correctly) over a narrow range of sequence similarity. Measuring sequence similarity by the HSSP-value (HVAL) (10,12) for the *daylight zone* at HVAL $>$ 5 ($>$25% PIDE - pairwise sequence identity over $>$250 aligned residues) over 90% of all protein pairs have similar 3D structures, while at the beginning of the *midnight zone* for HVAL$\leftarrow$5 ($<$15% PIDE for $>$ 250 aligned residues), over 90% have different 3D structures. Thus, the transition from daylight to midnight zone is described by a phase-transition in which over about ten percentage points in PIDE precision drops from 90% to 10%, i.e. from almost *all correct* to almost *all incorrect* within ±5 points PIDE. The particular point at which the twilight zone begins and how extreme the transition is, depends on the phenotype: steeper at lower PIDE for structure (10) and flatter at higher PIDE for function (13,14).

[*]To whom correspondence should be addressed. Email: mheinzinger@rostlab.org
Correspondence may also be addressed to Burkhard Rost. Tel: +49 289 17 811; Email: assistant@rostlab.org

If two proteins have highly similar structures, it is still possible for their sequences to be found in this *midnight zone*, i.e. have seemingly random sequence similarity (11). Thus, if we could safely lower the threshold just a little, we would gain many annotations of structural and functional similarity. In fact, any push a little lower reveals many proteins with similar phenotype, e.g. structure or function. Unfortunately, without improving the search method, such a lowering usually comes at the expense of even more proteins with dissimilar phenotype.

This simple reality has been driving the advance of methods using sequence similarity to establish relations: from advanced pairwise comparisons (15,16) over sequence-profile (17–20) to profile-profile comparisons (8,21,22,23,24,25,26) or efficient shortcuts to the latter (27,28). All those methods share one simple idea, namely, to use evolutionary information (EI) to create families of related proteins. These are summarized in multiple sequence alignments (MSAs). Using such information as input to machine learning methods has been generating essentially all state-of-the-art (SOTA) prediction methods for almost three decades (29–31). Using MSAs has also been one major key behind the breakthrough in protein structure prediction through *AlphaFold2* (32), and subsequently of *RoseTTAFold* (33) which builds on ideas introduced by *AlphaFold2*, i.e. allowing for communication between different sequence- and structure modules within the network. Transfer- or representation-learning offer a novel route toward comparisons of and predictions for single sequences without MSAs.

Embeddings capture language of life written in proteins

The introduction of LSTM- or attention-based Language Models (LMs) such as ELMo (34) or BERT (35) enabled a better use of large, unlabeled text corpora which arguably improved all tasks in natural language processing (NLP) (36). These advances have been transferred to proteins through protein Language Models (pLMs) equating amino acids with words in NLP and the sequence of entire proteins with sentences. Such pLMs learn to predict masked or missing amino acids using large databases of raw protein sequences as input (37–43), or by refining the pLM through another supervised task (44,45). Processing the information learned by the pLM, e.g. by using the output of the last hidden layers of the networks forming the pLMs, yields a representation of protein sequences referred to as embeddings (Figure 1 in (37)). Embeddings have been used successfully as exclusive input to predicting secondary structure and subcellular localization at performance levels almost reaching (38–40) or even exceeding (37,46,47) the SOTA using evolutionary information from MSAs as input. Embeddings can even substitute sequence similarity for homology-based annotation transfer (48,49). The power of such embeddings has been increasing with the advance of algorithms and the growth of data (37). The recent advances have shown that a limit to such improvements has not nearly been reached when writing this (22.02.2022).

Embeddings from pLMs capture a diversity of higher-level features of proteins, including various aspects of protein function and structure (37,38,40,48,49,50,51). In fact, pLMs such as ProtT5 (37) or ESM-1b (38) capture aspects about protein structure so impressively that inter-residue distances – and consequently 3D structure – can be predicted without using MSAs, even with relatively small (few free parameters) Deep Learning (DL) architectures (52).

Supervised learning directly maps the input to the class output. Instead, contrastive learning (53), optimizes a new embedding space in which similar samples are pushed closer, dissimilar samples farther apart. Contrastive learning relies only on the similarity between pairs (or triplets) of samples instead of on class label. The definition of similarity in embedding rather than sequence space, combined with contrastive learning, offered an alternative to sequence-based protein comparisons. This led us to hypothesize that we might find structurally and functionally consistent subgroups within protein families from raw sequences. As a proof-of-principle, a rudimentary precursor of this work helped to cluster FunFams (54,49). The benefit of optimizing embeddings specifically for SCOPe fold recognition (55) has recently been shown (44,50,56). Other approaches toward fold recognition deep learn fold-specific motifs (57), pairwise similarity scores (58) or sequence alignments (59). However, most of the top-performing solutions rely on information extracted from MSAs (60) and do not utilize the transfer-learning capabilities offered by recent pLMs.

Here, we expand on the hypothesis that replacing supervised learning by contrastive learning intrinsically fits the hierarchy of CATH (5,54). We propose an approach that marries both, self-supervised pretraining and contrastive learning, by representing protein sequences as embeddings, and using increasing overlap in the CATH hierarchy as a notion of increasing structural similarity to contrastively learn a new embedding space. We used the pLM ProtT5 (37) as static feature encoder (no fine-tuning of the pLM) to retrieve initial embeddings that were then mapped by a feed-forward neural network (FNN) to a new, learned embedding space optimized on CATH through contrastive learning. More specifically, the Soft Margin Loss was used with triplets of proteins (anchor, positive, and negative) to optimize the new embedding space toward maximizing the distance between proteins from different CATH classes (anchor-negative pairs) while minimizing the distance between proteins in the same CATH class (anchor-positive pairs). Triplets of varying structural similarity were used simultaneously to optimize a single, shared network: all four CATH-levels were simultaneously learned by one FNN. The resulting model was called *ProtTucker* and its embeddings were established to identify more distant relations than is possible from sequence alone. One important objective of *ProtTucker* is to study entire functional modules through identifying more distant relations, as found to be crucial for capturing mimicry and hijacking of SARS-CoV-2 (61).

METHODS

CATH hierarchy

The CATH (6,54) hierarchy (v4.3) classifies three-dimensional (3D) protein structures from the PDB (Protein Data Bank (62)) at the four levels Class, Architecture, Topology and Homologous superfamily. On average, higher levels (further away from root: H > T > A > C) are

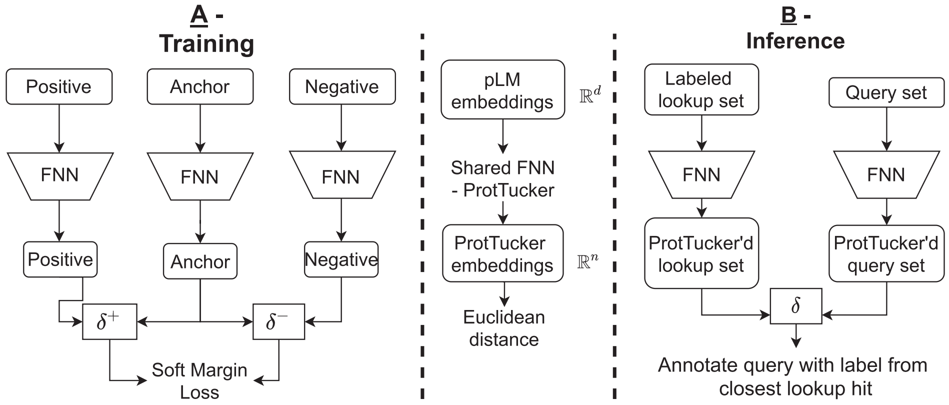

Figure 1. Sketch of ProtTucker. Panel **A** illustrates how protein triplets were used to contrastively learn the CATH hierarchy (5,54). First, protein Language Models (pLMs) were used as static feature encoders to derive embeddings for protein sequences (anchor, positive, negative). The embedding of each protein was processed separately by the same, shared FNN with hard parameter sharing, called ProtTucker. During optimization, the Soft Margin Loss was used to maximize the distance between proteins from different CATH classes (anchor-negative pairs) while minimizing the distance between proteins in the same CATH class (anchor-positive pairs). All four CATH-levels were simultaneously learned by the same FNN. This resulted in a newly, learned CATH-optimized embedding for each protein. Panel **B** sketches how the contrastive learning FNN is used for prediction of new proteins (inference). For all proteins in a lookup set with experimental annotations (labeled proteins; here the CATH lookup set), as well as for a query protein without experimental annotations (unlabeled proteins) all embeddings are extracted in two steps: (1) extract per-residue embeddings from original pLM and create per-protein embeddings by averaging over protein length. (2) Input those embeddings into the pre-trained FNNs, i.e. ProtTucker. Similar to homology-based inference (HBI), predictions are generated by transferring the annotation of the closest hit from the lookup set to the query protein. The embedding-based annotation transfer (EAT) transferred annotations to the hit with the smallest Euclidean distance in *ProtTucker* embedding space.

more similar in their 3D structure or have more residues for which the same level of 3D similarity is reached. We used increasing overlap in this hierarchical classification as a proxy to define increasing structural similarity between protein pairs. For example, we assumed that any two proteins with the same topology (T) are structurally more similar than any two proteins with identical architecture (A) but different topology (T). In more formal terms: SIM3D(P1,P2)>SIM3D(P3,P4), where T(P1) = T(P2) & T(P3)≠T(P4) & A(P3) = A(P4). This notion of similarity was applied on all four levels of CATH.

Data set

The sequence-unique datasets provided by CATH (5,54) v4.3 (123k proteins, CATH-S100) provided training and evaluation data for ProtTucker. A test set (300 proteins, dubbed test300 in the following) for final evaluation and a validation set (200 proteins, dubbed *val200*) for early stopping were randomly split off from CATH-S100 while ensuring that (1) every homologous superfamily appeared maximally once in test300 ∩ val200 and (2) each protein in test300 & val200 has a so called Structural Sub-group (SSG) annotation, i.e. clusters of domain structure relatives that superpose within 5Å (0.5 nm), in CATH. To create the training set, we removed any protein from CATH-S100 that shared more than 20% pairwise sequence identity (PIDE) to any validation or test protein according to MMSeqs2 (27) applying its iterative profile-search (*–num-iterations 3*) with highest sensitivity (*-s 7.5*) and bidirectional coverage (*–cov-mode 0*). Additionally, large families (>100 members) within CATH-S100 were clustered at 95% PIDE and length coverage of 95% of both proteins using MMSeqs2 (bidirectional coverage; *–cov-mode 0*). The cluster representatives were used for training (66k proteins, dubbed *train66k*) and as lookup set during early stopping on set *val200*. We needed a lookup set from which to transfer annotations because contrastive learning outputs embeddings instead of class predictions. For the final evaluation on *test300*, we created another lookup set but ignored val200 proteins during redundancy reduction (69k proteins, dubbed *lookup69k*). This provided a set of proteins for validation which had similar sequence properties to those during the final evaluation while 'hiding' them during training and not using them for any other optimization. To ensure strict non-redundancy between lookup69k and test300, we further removed any protein from *test300* with HVAL > 0 (10) to any protein in *lookup69k* (219 proteins, dubbed *test219* in the following). All performance measures were computed using *test219*.

Data augmentation can be crucial for contrastive learning to reach performance in other fields (63). However, no straightforward way exists to augment protein sequences as randomly changing sequences very likely alters or even destroys protein structure and function. Therefore, we decided to use homology-based inference (HBI) for data augmentation during training, i.e. we created a new training set based on Gene3D (64) (v21.0.1) which uses Hidden Markov Models (HMMs) derived from CATH domain structures to transfer annotations from labeled CATH to unlabeled UniProt. We first clustered the 61M protein sequences in Gene3D at 50% PIDE and 80% coverage of both proteins (bidirectional coverage; *–cov-mode 0*) and then applied the same MMSeqs2 profile-search (*–num-iterations 3 –s 7.5*) as outlined above to remove cluster representatives with ≥ 20% PIDE to any protein in test300 or val200 (PIDE(P_{train},$P_{\text{test300|val200}}$) ≤ 20%). This filtering yielded 11M sequences for an alternative training set (dubbed *train11M*).

The CATH detection of ProtTucker was further analyzed using a strictly non-redundant, high-quality dataset. This set was created by first clustering CATH v4.3 at 30% using HMM profiles from HMMER and additionally discarding all proteins without equivalent entry in SCOPe, i.e. the domain boundaries and the domain-superfamily assignment had to be nearly identical (3186 proteins, CATH-S30). We used the highly sensitive structural alignment scoring tool SSAP (65,66) to compute the structural similarity between all protein pairs in this set.

We probed whether ProtTucker embeddings might also help in solving tasks unrelated to protein structure/CATH, using as proxy a dataset assessing subcellular location prediction in ten states (46,67). We embedding-transferred annotations (EAT) from the standard *DeepLocTrain* set to 490 proteins in a recently proposed test set (*setHard*) that was strictly non-redundant to *DeepLocTrain*. Datasets described elsewhere in more detail (46,67). Finally, we showcased predictions for entire organisms using three UniProt reference proteome: *Escherichia coli* (E. Coli; reviewed, Swiss-Prot (68)), *Armillaria ostoyae* (A. ostoyae; unreviewed, TrEMBL (68)) and *Megavirus Chilensis* (M. Chilensis; unreviewed TrEMBL (68)).

Data representation

Protein sequences were encoded through distributed vector representations (embeddings) derived from four different pre-trained protein language models (pLM): (–) **ProtBERT** (37) based on the NLP (Natural Language Processing) algorithm BERT (35) but trained on BFD (Big Fantastic Database) with over 2.3 billion protein sequences (69). (2) **ESM-1b** (38) is similar to (Prot)BERT but trained on UniRef50 (68). (3) **ProtT5-XL-U50** (37) (*ProtT5* for simplicity) based on the NLP sequence-to-sequence model T5 (70) trained on BFD and fine-tuned on Uniref50. (4) **ProSE** (44) trained long short-term memory cells (LSTMs) either solely on 76M unlabeled sequences from UniRef90 (**ProSE-DLM**) or on additionally predicting intra-residue contacts and structural similarity from 28k SCOPe proteins (55) (multi-task: **ProSE-MT**). While ProSE, ProtBert and ESM-1b were trained on reconstructing corrupted tokens/amino acids from non-corrupted (protein) sequence context (masked language modeling), ProtT5 was trained by *teacher forcing*, i.e. input and targets were fed to the model with inputs being corrupted protein sequences and targets being identical to inputs but shifted to the right (span generation with span size of 1 for ProtT5). Except for ProSE-MT, all pLMs were optimized only through self-supervised learning exclusively using unlabeled sequences for pre-training.

pLMs output a single vector for each residue yielding an $L \times N$-dimensional matrix (L: protein length, N: embedding dimension; $N = 1024$ for ProtBERT/ProtT5; $N = 1280$ for ESM-1b; $N = 6165$ for ProSE). From this $L \times N$ embedding matrix, we derived a fixed-size N-dimensional vector representing each protein by averaging over protein length (Figure 1 (37)). The pLMs were used as static feature encoder only, i.e. no gradient was backpropagated for fine-tuning. As recommended in the original publication (37), for ProtT5, we only used the encoder part of ProtT5 in half-precision to embed protein sequences. Similarly, ProtBERT embeddings were derived in half-precision.

Contrastive learning: architecture

A two-layer feedforward neural network (FNN) projected fixed-size per-protein (sentence-level) embed-dings from 1024-d (1280-d/6165-d for ESM-1b and ProSE respectively) to 256 and further to 128 dimensions with the standard hyperbolic tangent (*tanh*) as non-linearity between layers. We also experimented with deeper/more sophisticated networks without any gain from more free parameters (data not shown). This confirmed previous findings that simple networks suffice when inputting advanced embeddings (37,38,52,71). As the network was trained using contrastive learning, no final classification layer was needed. Instead, the 128-dimensional output space was optimized directly.

Contrastive learning: training

During training, the new embedding space spanned by the output of the FNN was optimized to capture structural similarity using triplets of protein embeddings. Each triplet had an anchor, a positive and a negative. In each epoch, all *train66k* proteins were used once as anchor, while positives and negatives were sampled randomly from *train66k* using the following *hierarchy-sampling*. First, a random level α (α = [1,2,3,4]) describing the increasing structural overlap between triplets was picked. This defined a positive (same CATH-number as anchor up to level α'≤α) and a negative label (same CATH-number as anchor up to level α'< α). For instance, assume the anchor has the CATH-label 1.25.10.60 (Rad61, Wapl domain) and we randomly picked α = 3 (topology-level), only proteins with the anchor's topology (1.25.10.x; Leucine-rich Repeat Variant) qualify as positives while all negatives share the anchor's architecture (1.25.y.z; Alpha Horseshoe) with different topology (y≠10). Self-hits of the anchor were excluded. From the training proteins fulfilling those constraints, one positive and one negative were picked at random. If no triplets could be formed (e.g. α = 4 with a single-member homologous superfamily had no positive for this anchor/α combination), α was changed at random until a valid triplet could be formed (eventually, all proteins found a class-level partner). This flexibility in sampling enabled training on all proteins, independent of family size.

Unlike randomly sampling negatives, the hierarchical sampling could be described as semi-hard sampling as it zoomed into triplets that were neither too easy (little signal) nor too hard (outliers) to classify by ensuring a minimal overlap between the anchor and the chosen negative/positive pair. Thereby, trivial triplets are undersampled (avoided), i.e. those with 3D structures so different that the separation becomes trivial (*daylight zone*). As the final triplet selection was still random, anchor-positive pairs could still be too easy/similar which was shown to hinder the success of contrastive learning (72). To solve this issue, we paired hierarchy-sampling with so called *batch-hard* *sampling* (72) which offers a computationally efficient solution for sampling *semi-hard* triplets within one mini-batch. More specifically, we combined *batch-hard sampling* with the triplets created using *hierarchy-sampling* by re-wiring all proteins, irrespective of anchor, positive or negative, within one mini-batch such that they satisfied the hierarchy-sampling criterion and had maximum/minimal Euclidean distance for anchor-positive/anchor-negative pairs. Sampling hard triplets only within each mini-batch instead of across the entire data set avoided extreme outliers (potentially too hard/noisy) while increasing the rate of semi-hard anchor-positive/anchor-negative pairs. Assume multiple proteins of the topology Leucine-rich Repeat Variant were within one mini-batch, the hardest positive for each anchor would be picked by choosing the anchor-positive pair with the largest Euclidean distance. Accordingly, anchor-negative pairs would be picked based on the smallest Euclidean distance. For each mini-batch, this sampling was applied to all four levels of the CATH-hierarchy, so triplets were re-wired on all four CATH levels resulting in a total batch-size of about: batch_size * 3 * 4. This was an 'about' instead of 'equal' because for some mini-batches, not all proteins had valid triplets for all four levels.

Finally, the same two-layer FNN was used (hard parameter sharing) to project the 1024-d (or 1280-d/6165-d for ESM-1b or ProSE respectively) embeddings of all proteins, irrespective of anchor, positive or negative, to a new 128-d vector space. The *Soft Margin Loss* was used to optimize this new embedding space such that anchor-positive pairs were pulled together (reduction of Euclidean distance) while pushing apart anchor-negative pairs (increase of Euclidean distance). The efficiency of combining the Soft Margin Loss with batch-hard sampling was established before (72), although without prior hierarchical triplet sampling. Here, we used Soft Margin Loss as implemented in PyTorch:

$$Loss\ (d, y) = \sum_{t} \frac{\log\left(1 + e^{(-y[t]*d[t])}\right)}{|d|} \qquad (1)$$

$$d = \left\{ \min_{\substack{n \in B \wedge a \neq n \\ C_i(a) \neq C_i(n)}} D(f(a), f(n)) - \max_{\substack{n \in B\ p \neq a \\ C_i(a) \neq C_i(p)}} D(f(a), f(p)) \right\} \qquad (2)$$

$$\forall a \in \{B\} \wedge \forall i \in 1, 2, 3, 4 \wedge \forall C_i \in \{CATH\,labels\}$$

B represents one mini-batch created through hierarchical sampling, *f(a)*, *f(p)* and *f(n)* represent the *ProtTucker* embeddings of proteins *a* (anchor), *n* (negative), and *p* (positive) represented as pLM embeddings; C_i represents the CATH annotation of a protein on the *i*'th hierarchy level of CATH; finally, D(f(a),f(x)) represents the Euclidean distance between the embeddings for proteins a and x. We created the mini-batch *B* used for training by choosing for each protein or anchor *a* in *B* the hardest negative *n* and the hardest positive *p* by picking those proteins in *B* that have the smallest | largest Euclidean distance *D* to *a ProtTucker* embedding space while not sharing | sharing C_i, respectively. Those semi-hard triplets are indexed by *t* and *d[t]* referring to the difference between *D* of anchor-negative and *D* of anchor-positive. In our case, the label for the *t*'th triplet *y[t]* is always 1 as the sign of x indicates training success, i.e. whether the distance anchor-positive is smaller than that between anchor-negative.

Consequently, triplets of varying structural similarity were used simultaneously to optimize a single, shared network, i.e. all four CATH-levels were learned by the same network at the same time (Figure 1A). We used the *Adam optimizer* (73) with a learning rate of 0.001, and a batch-size of 256 to optimize the network. The effective batch-size increased due to batch-hard sampling to a maximum of 3072, depending on the number of valid triplets that could be formed within the current mini-batch. Training terminated (*early stopping*) at the highest accuracy in predicting the correct homologous superfamily for set *val200*.

Evaluation and prediction (inference)

While supervised training directly outputs class predictions, contrastive learning, outputs a new embedding space. Thus, predictions (inferences) were generated as for homology-based inference (HBI), i.e. given protein X with experimental annotation (CATH assignment) and a query protein Q without, then HBI transfers the annotation from X to Q if sequence similarity SIM(X,Q)> threshold. For contrastive learning, we replaced SIM(X,Q) by D(f(X),f(Q)) with D as the shortest Euclidean distance in embedding space (Figure 1B). In previous studies (37,48,49), we found the Euclidean distance performed better than the cosine distance which is more common in AI/NLP. The Euclidean distance also optimized the ProtTucker embeddings. Set *test219* with the *lookup69k* as lookup set (set of all X) served as final evaluation. If no protein in the lookup set shared the annotation of the query protein at a certain CATH-level (more likely for H than for C), the sample was excluded from the evaluation of this CATH-level as no correct prediction was possible (Supplementary Table S1).

During evaluation, we compared the performance of our embedding-based annotation transfer (EAT) to HBI using the sequence comparisons from MMSeqs2 (27). While transferring only the HBI hit with the lowest E-value, we searched for hits up to an E-value of 10 to ensure that most proteins had at least one hit while using the highest sensitivity setting (-s 7.5). Additionally, we used publicly available CATH-Gene3D (54) hidden Markov Models (HMMs) along with HMMER (74) to detect remote homologs up to an *E*-value of 10.

For both approaches, EAT and HBI, we computed the accuracy as the fraction of correct hits for each CATH-level. A hit at lower CATH-levels could be correct if and only if all previous levels were correctly predicted. Due to varying number of samples at different CATH-levels (Supplementary Table S1), performance measures not normalized by background numbers could be higher for lower levels. Predictions were counted as incorrect if a query did not have a hit in the lookup set but a lookup protein of the same CATH annotation existed. This not only affected the number of proteins available at different CATH-levels (Supplementary Table S2) but also the number of classes (Supplementary Table S3). A random baseline was computed by transferring annotations from a randomly picked protein in *lookup69k* to *test219*.

Performance measures

The four coarse-grained classes at the top CATH level (‘C’) are defined by their secondary structure content. These four branch into 5481 different superfamilies with distinct structural and functional aspects (CATH v4.3.0). However, most standard metrics are defined for binary cases which requires some grouping of predictions into four cases: 1) TP (true positives): correctly predicted to be in the *positive* class, 2) TN (true negatives): correctly predicted to be in the *negative* class, 3) FP (false positives): incorrectly predicted to be positives, and 4) FN (false negatives): incorrectly predicted to be in in the negative class. Here, we focused on performance measures applicable for multiclass problems and are implemented in scikit (75). These were in particular: **accuracy** (Acc, Equation 3) as the fraction of correct predictions

$$Accuracy\ (y, \hat{y}) = \frac{1}{n\_samples} \sum_{i\,=\,0}^{n\_samples-1} 1\,(\,\widehat{y_i} = y_i\,)\ (3)$$

with y_i being the ground truth (experimental annotation) and $\widehat{y_i}$ the prediction for protein i. In analogy, we defined **coverage** as the proportion of the *test219* proteins for which a classifier made a prediction at a given prediction reliability $\widehat{y_i^r}$ and reliability threshold θ:

$$Coverage\,(y, \hat{y}) \;=\; \frac{1}{n\_samples} \sum_{i\,=\,0}^{n\_samples-1} 1(\hat{y}_i^r\ lt;\theta)\ (4)$$

In these definitions accuracy corresponds to precision, and coverage to recall binarizing a multiclass problem through micro-averaging, i.e. by counting the total TPs, FPs and FNs globally, irrespective of the class. The multi-class extension of Matthew’s correlation coefficient (MCC, (31)) was defined as:

$$MCC \;=\; \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{\left(s^2 - \sum_k^K p_k^2\right) \times \left(s^2 - \sum_k^K t_k^2\right)}} \qquad (5)$$

with $t_k = \sum_i^K C_{ik}$ as the number of times class k truly occurred, $p_k = \sum_i^K C_{ki}$ as the number of times class k was predicted, $c \;= \sum_k^K C_{kk}$, the total number of samples correctly predicted, and $s \;= \sum_i^K \sum_j^K C_{ij}$, the total number of samples.

95% confidence intervals for accuracy and MCC were estimated over $n = 1000$ bootstrap sets; for each bootstrap set we randomly sampled predictions from the original test set with replacement. Standard deviation (or in the case of bootstrapping: bootstrap standard error) was calculated as the difference of each test set (x_i) from the average performance $\langle X \rangle$ (Equation 6). 95% confidence intervals were estimated by multiplying the bootstrap standard error by 1.96.

$$\text{StdDev} \;= \sqrt{\frac{x_i - \langle X \rangle^2}{n}} \qquad (6)$$

RESULTS

Generalization of HBI to EAT

Homology-based inference (HBI) uses sequence similarity to transfer annotations from experimentally characterized (labelled) to uncharacterized (unlabeled) proteins. More specifically, an unlabeled query protein Q is aligned against a set of proteins X with experimental annotations (dubbed *lookup set*) and the annotation of the best hit, e.g. measured as lowest E-value, is transferred if it is below a certain threshold (e.g. E-value(Q,X)<10–3). This relates to inferring the annotation of a query protein from the *k*-Nearest Neighbors (k-NN) in sequence space. More recently, similar approaches expanded from distance in sequence to distance in embedding space (Figure 1B) as means for k-NN based annotation transfer (48,50). Here, we refer to such methods as embedding- based annotation transfer (EAT). We used EAT as a proxy for the comparison of embeddings from five different pLMs: ProSE-DLM & ProSE-MT (44), ProtBERT & ProtT5 (37), and ESM-1b (38). Next, we used triplets of proteins (anchor, positive, negative) to learn a new embedding space by pulling protein pairs from the same CATH class (anchor-positive) closer together while pushing apart pairs from different CATH classes (anchor-negative; Figure 1A). We referred to this method as *ProtTucker*. At this stage, we did not fine-tune the pre-trained pLMs. Instead, we created a new embedding space using a two-layer feed-forward neural network (FNN).

EAT with raw embeddings level with HBI

First, we transferred annotations from all proteins in *lookup69k* to any protein in *test219*. All pLMs significantly (at 95% CI—confidence interval) outperformed random annotation transfer (Table 1). Performance differed between pLMs (Table 1), with ProtBERT (37) being consistently worse than LSTM-based ProSE-DLM or more advanced transformers (ESM-1b, ProtT5). ESM-1b and ProtT5 also numerically outperformed ProSE-DLM and HBI using MMseqs2 (27), especially on the most fine-grained and thus hardest level of superfamilies. However, MMseqs2 had been used for redundancy-reduction, i.e. the data set had been optimized for minimal performance of MMseqs2. HBI using publicly available HMM-profiles from CATH-Gene3D (54) along with the profile-based advanced HMMER (74) designed for more remote homology detection, outperformed all raw embeddings for homologous superfamilies, while embeddings from ESM-1b and ProtT5 appeared superior on the class- and architecture-level (Table 1). In fact, all HBI values, for MMseqs2 and HMMER, were highest for the H-level, and second highest for the C-level. In contrast, raw pLM embeddings mirrored the random baseline trend, with numbers being inversely proportional to the rank in C, A, T, H (highest for C, lowest for H, Table 1).

Table 1. Accuracy for annotation transfer to queries in test219 *

	Method/Input	C	A	T	H	Mean
Baseline	Random	29 ± 6	9 ± 4	1 ± 2	0 ± 0	10 ± 3
HBI	MMSeqs2 (sequence)	52 ± 7	36 ± 6	29 ± 6	35 ± 6	38 ± 6
	HMMER (profile)	70 ± 6	60 ± 6	59 ± 7	77 ± 7	67 ± 6
EAT - unsupervised	ProSE-DLM	74 ± 6	48 ± 7	28 ± 6	25 ± 7	44 ± 6
	ESM-1b	79 ± 5	61 ± 6	50 ± 7	57 ± 8	62 ± 7
	ProtBERT	67 ± 6	38 ± 6	22 ± 6	18 ± 6	36 ± 6
	ProtT5	84 ± 5	67 ± 6	57 ± 6	64 ± 8	68 ± 6
EAT - supervised	ProSE-MT	82 ± 5	65 ± 6	52 ± 7	56 ± 8	64 ± 7
EAT - contrastive learning—ProtTucker	ProSE-DLM	78 ± 4	53 ± 6	32 ± 6	29 ± 7	48 ± 6
	ProSE-MT	87 ± 4	68 ± 6	53 ± 7	55 ± 8	66 ± 6
	ESM-1b	87 ± 4	68 ± 6	59 ± 7	70 ± 7	71 ± 6
	ProtBERT	81 ± 5	52 ± 7	37 ± 6	39 ± 8	52 ± 7
	ProtT5	**89 ± 4**	75 ± 6	64 ± 6	76 ± 6	76 ± 6
	ProtT5 (train11M)	88 ± 4	**77 ± 5**	**68 ± 5**	**79 ± 7**	**78 ± 6**

*Accuracy (Equation 3) for predicting CATH (54,1) levels (C, A, T, H) by transferring annotations from *Lookup69k* (lookup set) to *test219* (queries); shown for each of the four levels from the most coarse-grained level class C to the most fine-grained level of homology H. The column *Mean* marked the simple arithmetic average over the four performance values. Queries with at least one lookup protein of the same CATH classification but without any hit at E-value < 10 for MMSeqs/HMMER were counted as incorrect predictions. Errors indicate bootstrapped 95% confidence intervals, i.e. 1.96 bootstrap standard errors (Equation 6). Queries with at least one lookup protein of the same CATH annotation but without any hit (no hit with E-value < 10 for MMSeqs/HMMER; irrelevant for EAT) were counted as wrong predictions. **Bold** letters mark the numerically highest values (averages over all *test219* proteins) in each column irrespective of the confidence interval.

Methods: Baseline: Random transferred the label of a randomly picked protein; HBI: MMSeqs2 (27) used single sequence search to transfer the annotation of the hit with the lowest *E*-value; HBI: HMMER used HMM-profiles (74); EAT-unsupervised: embedding-based transfer of annotations using the smallest Euclidean distance measured in embedding space of unsupervised pLMs ProSE-DLM, ESM-1b (38), ProtBERT and ProtT5 (37); EAT-supervised: annotation transfer using ProSE-MT trained on structural data in SCOPe; EAT: contrastive learning ProtTucker: contrastive learning trained on CATH classifications in *train66k* using as input embeddings from ProSE-DLM, ProSE-MT, ESM-1b, ProtBERT and ProtT5; ProtTucker-ProtT5 (train*11M*) trained on additional data from Gene3D (*train11M*).

EAT improved through supervised embeddings

ProSE-MT expands ProSE-DLM by additionally training on intra-residue contacts and structural similarity using labeled data from SCOPe (44). This additional effort was reflected by the higher performance for all CATH levels (Table 1, ProSE-MT > ProSE-DLM). The supervision pushed the LSTM-based ProSE-MT to reach performance levels close to the unsupervised, raw embeddings from transformer-based ProtT5. The performance gap increased with classification difficulty (Table 1, ProtT5 > ProSE-MT, especially at the H-level).

EAT improved by contrastively learning embeddings

Contrastive learning tries to bring members from the same class/CATH-level closer while pushing those from different classes further apart. One success is the degree to which these two distributions (same versus different) were separated through training: the distribution of all pairwise Euclidean distances within (intra/same) and between (inter/different) superfamilies in *train66k* changed substantially through contrastive learning (Figure 2). Before applying contrastive learning, the distributions between (inter, Figure 2: red) and within (intra, Figure 2: blue) overlapped much more than after.

Displaying the information learned by the embeddings, we compared t-SNE projections colored by the four main CATH classes before (Figure 3A) and after (Figure 3C) contrastive learning. These two projections compared 1024 dimensions from ProtT5 (Figure 3A) with 128 dimensions from ProtTucker (Figure 3C). To rule out visual effects from higher dimensionality, we also compared the untrained, randomly initialized version of ProtTucker using pre-trained ProtT5 embeddings as input (Figure 3B). For all cases, the data set (*train66k*) and the parameters for dimensionality reduction were identical. T-SNE projections of raw ProtT5 embeddings qualitatively suggested some class separation (clustering). The information underlying this separation was preserved when projecting ProtT5 embeddings through an untrained ProtTucker (Figure 3B). Embeddings from ProtTucker(ProtT5), i.e. those resulting through contrastive learning, separated the classes much more clearly.

To further probe the extent to which contrastive learning captured remote homologs, we compared the Euclidean distance between all protein pairs in a 30% non-redundant dataset (CATH-S30) with the structural similarity of those pairs computed via SSAP (65,66) (Figure 4). From the ∼10M possible pairs between the 3,186 proteins in CATH-S30 (problem not fully symmetric, therefore N*(N - 1): 10.1M), 7.1M had to be discarded due to low quality (SSAP-score < 50), leaving 2.9M pairs of which only 1.8% (53k pairs) had the same homologous superfamily (Figure 4: blue). Despite this imbalance, unsupervised ProtT5 (Figure 4A) already to some extent separated the same from different superfamilies. Still, ProtTucker(ProtT5) improved this separation, especially, for pairs with low structural similarity (Figure 4B). This was supported by the Spearman correlation coefficient between the structural similarity and the Euclidean distance increasing from 0.05 to 0.22 after contrastive learning. When

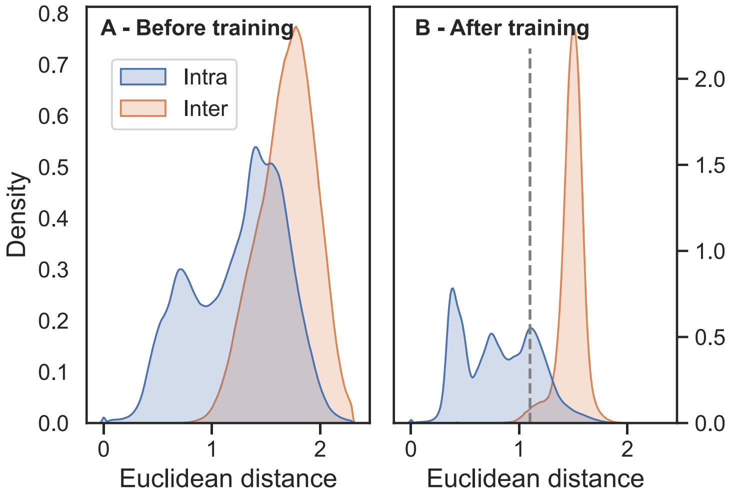

Figure 2. Contrastive learning separated positives from negatives. The structural similarity defined by increasing overlap in CATH drove the contrastive learning of a new embedding space. The new embeddings distanced protein pairs with different CATH classifications (red; same topology but different superfamily) while focusing pairs with the same CATH classification (blue; same superfamily). These graphs compared the Euclidean distance for all such pairs from the set *train66k* before (Panel **A**) and after (Panel **B**) contrastive training. Input to the FNN were the raw embeddings from ProtT5 (37), output were the new ProtTucker(ProtT5) embeddings. The dashed line at Euclidean distance of 1.1 in B marked the threshold at which EAT performances started to decrease (Figure 5).

considering only the subset of pairs that likely have similar folds (SSAP-score > 70), this correlation increased to 0.26 and 0.37 for ProtT5 and ProtTucker(ProtT5), respectively.

The trend captured by the better separation of distributions (Figure 2) and structural features (Figures 3 and 4) translated directly into performance increases: all embeddings optimized on the CATH hierarchy through contrastive learning yielded better EAT classifications than the raw embeddings from pre-trained pLMs (Table 1). As ProtTucker described the process of refining raw embeddings through contrastive learning, we used the annotation ProtTucker(*X*)—in this section also shortened to PT(X)—to refer to the embeddings output by inputting the pre-trained pLM *X* into the contrastive learning. The improvements were larger for more fine-grained CATH levels: all models improved significantly for the H-level, while only PT(ProtBERT) and PT(ESM-1b) improved from 4 to 14 or from 0 to 21 percentage points for the C-, and the H-level, respectively. PT(ProtT5) consistently outperformed all other pLMs on all four CATH-levels, with an increasing performance gap toward the more fine-grained H-level at which all pLMs except for PT(ESM-1b) performed significantly worse. The improvements from contrastive learning for PT(ProSE-DLM) and PT(ProSE-MT) were mostly consistent but largely insignificant. Especially, the model already optimized using labeled data (ProSE-MT) hardly improved through another round of supervision by contrastive learning and even worsened slightly at the H-level.

We augmented the training set for PT(ProtT5) by adding HBI-hits from HMM-profiles provided by CATH-Gene3D (if sequence dissimilar to test300/val200). This increased the training set from 66k (66 × 10^3) to 11m (11 × 10^6) proteins (15-fold increase) and raised performance, although the higher values were neither statistically significant nor consistent (Table 1: values in last row not always higher than those in second to last row).

Ablation study

We studied the effect of *batch-hard* and *hierarchical* sampling on performance by removing each component when training PT(ProtT5) (Table 2). Benchmarking on EAT from *lookup69k* to *test219* established that removing either component lowered performance for all CATH levels. Dropping both sampling methods substantially lowered performance. While dropping *batch-hard* sampling still reached high performance for the coarse-grained C- and A-level, dropping hierarchy-sampling dropped performance for both. Dropping both sampling technique, performed worse for all CATH levels but the decrease for the more fine-grained superfamily level was much larger than for the C-level.

Embedding distance correlated with accuracy

The MCC, (Equation 5) of HBI inversely correlated with *E*-value (Figure 6, HBI-methods): more significant hits (lower E-values) more often shared the same CATH level than less significant hits (higher *E*-values). In analogy, we explored the corresponding relation for EAT, namely the correlation between accuracy (Equation 3) and embedding distance for ProtTucker(ProtT5). Indeed, accuracy correlated with embedding distance (Figure 5: solid lines) while recall inversely correlated (Figure 5: dashed lines) for all four classes. For instance, when transferring only annotations for closest hits

Table 2. Ablation study[*]

	C	A	T	H	Mean
Baseline	89 ± 4	75 ± 6	64 ± 6	76 ± 6	76 ± 6
-batch-hard	88 ± 4	73 ± 6	62 ± 7	69 ± 7	73 ± 6
-hierarchy	83 ± 5	69 ± 6	62 ± 7	71 ± 7	71 ± 6
-both	83 ± 5	63 ± 6	51 ± 7	57 ± 8	64 ± 7

[*] Accuracy (Equation 3) and 95% CI (Equation 6) for predicting CATH-levels (54,1) through EAT from *Lookup69k* (lookup set) to *test219* (queries). We investigate the effect on performance when dropping either *batch-hard* sampling, *hierarchy*-sampling or *both* from the *Baseline* model (ProtTucker(ProtT5)).

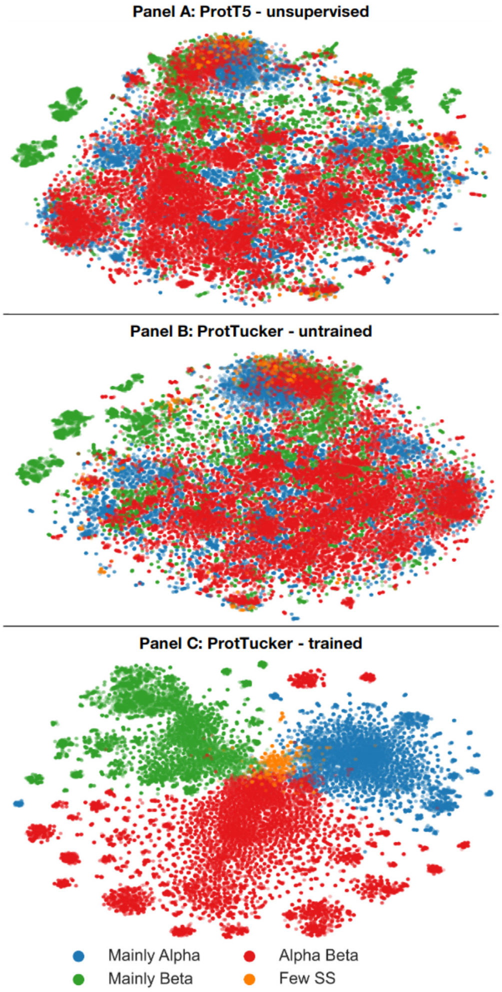

Figure 3. Better CATH class-level clustering. Using t-SNE (86), we projected the high-dimensional ProtTucker(ProtT5) embedding space onto 2D *before* (Panel **A**; ProtT5) and *after* (Panel **C**; ProtTucker(ProtT5)) contrastive learning. Panel **B** visualized the same data embedded with an untrained version of ProtTucker to assess the impact of different embedding dimensions (1024-d for ProtT5 versus 128-d for ProtTucker(ProtT5)). For all plots, dimensionality was first reduced by Principal Component Analysis (PCA) to 50 dimensions and parameters of the subsequent t-SNE were identical (perplexity = 150, learning_rate = 400, n_iter = 1000, seed = 42). The colors mark the major class level of CATH (C), distinguishing proteins according to their major distinction in secondary structure content.

with Euclidean distances of 1.1 or less, predictions were made for 57%, 57%, 59% or 75% of the test set (coverage, Equation 4) of these 96%, 93%, 91% or 90% were correct for levels C, A, T, H, respectively.

ProtTucker reached into the midnight zone

Annotation transfer by HBI crucially depends on the sequence similarity between query (unknown annotation) and template (experimental annotation). Usually, the significance of an inference is measured as the chance of finding a hit at random for a given database size (*E*-value; the lower the better). Here, we compared the effect of gradually removing hits depending on their *E*-values. Essentially, this approach measured how sensitive performance was to the degree of redundancy reduction between query and lookup set. For instance, at a value of 10^{-3} (dashed vertical lines in Figure 6), all pairs with *E*-values $\leq 10^{-3}$ were removed. HBI based on sequence alone performed much better with than without residual redundancy (Figure 6). The trend was similar for EAT, but much less pronounced: EAT succeeded for pairs with very different sequences (Figure 6 toward right) almost as well as for proteins with more sequence similar matches in the set (Figure 6 toward left: EAT almost as high as toward right).

ProtTucker not a generalist

We evaluated the generality of ProtTucker embeddings by (mis)-using them as exclusive input to predict subcellular location in ten states. To this end, we EAT transferred annotations from an established data set (Supplementary Table S5) to a strictly non-redundant test set (*setHard*, Supplementary Table S5). ProtTucker(ProtT5) embeddings outperformed the raw ProtT5 embeddings in the CATH classification for which they were optimized (structural similarity; Table 1), there appeared no performance gain in predicting location. Conversely, performance also did not decrease significantly, indicating that the new embeddings retained some of the information available in ProtT5 embeddings.

Family size mattered

By clustering very large protein families (>100 members after redundancy reduction) at 95% PIDE, we constrained the redundancy in set *train66k*. Nevertheless, when splitting *test219* into three bins of varying family sizes, we still observed a trend towards higher accuracy (Equation 3) for larger families at the H-level (Supplementary Figure S1). We chose the three bins such that they contained about

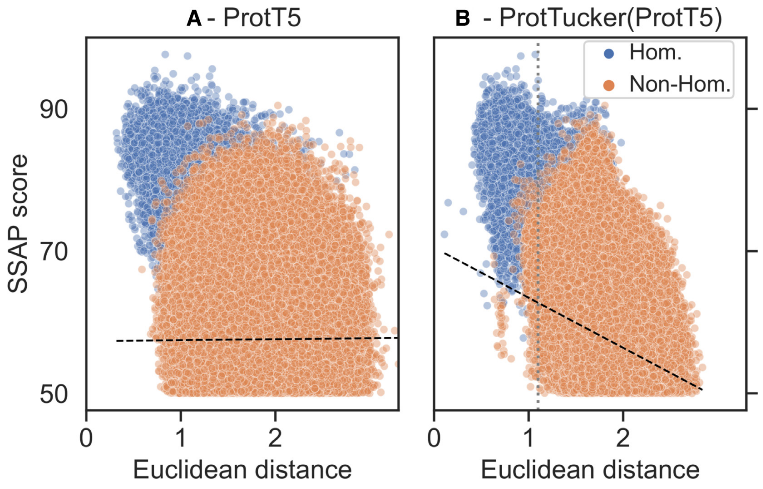

Figure 4. ProtTucker captured fine-grained structural similarity. 3186 non-redundant proteins (CATH-S30) probed the remote homology detection of embeddings *before* (Panel **A**) and *after* contrastive learning (Panel **B**). The Euclidean distance between ProtTucker embeddings (Panel B) correlated better with structural similarity computed by SSAP (65,66) than unsupervised embeddings (Panel A): Spearman ρ = 0.22 and ρ = 0.05 (black dashed lines). This correlation increased to ρ = 0.37 and ρ = 0.26 for structurally more similar protein pairs (SSAP-score > 70). Only 1.8% (53k) of all structurally similar pairs were in the same homologous superfamily (blue). The unsupervised ProtT5 already separated homologous pairs from others, but ProtTucker(ProtT5) improved, especially, for hard cases with low structural similarity. The gray dashed line at Euclidean distance = 1.1 in Panel B marked the threshold at which EAT performances started to decrease (Figure 5).

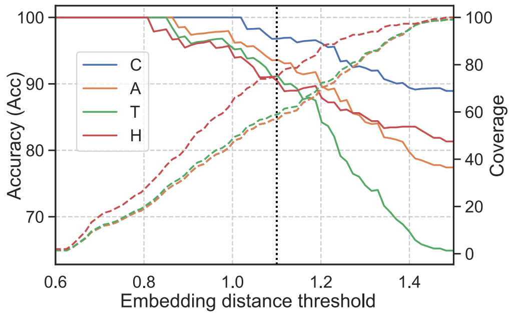

Figure 5. Embedding distance correlated with reliability. Similar to varying *E*-value cut-offs for HBI, we examined whether the fraction of correct predictions (accuracy; left axis; Equation 3) depended on embedding distance (x-axis) for EAT. This was shown by transferring annotations for all four CATH levels (Class: blue; Architecture: orange; Topology: green; Homologous superfamily: red) from *lookup69k* to the queries in set *test219* (Panel **B** in Figure 1) using the hit with lowest Euclidean distance. The fraction of *test219* proteins having a hit below a certain distance threshold (coverage, right axis, dashed lines; Equation 4) was evaluated separately for each CATH level. For example, at an Euclidean distance of 1.1 (vertical dotted line), 75% of the proteins found a hit at the *H*-level (Cov(H) = 75%) and 90% were correctly predicted (Acc(*H*) = 90%; SOM Tables S3 and S4 for more details). Thus, decreasing embedding distance correlated with EAT performance. This correlation enables users to select only the, e.g. 10% top hits, or as many hits to a certain CATH level as possible, depending on the objectives.

the same number of samples (small families: ≤10 members, medium: 11–70, and large: ≥70 members). Especially, unsupervised EAT using the raw ProtT5 embeddings exhibited a clear trend towards higher accuracy with increasing family size. In contrast, the two HBI-methods (MMseqs2, HMMER), as well as EAT using the optimized ProtTucker(ProtT5) embeddings performed similarly for small and medium-sized families and much better for large families.

EAT complements HBI

As previously shown (49), ProtTucker can improve clustering functional families (76). Here, we showed how EAT can be used to detect outliers. Firstly, we computed pairwise Euclidean distances between the embeddings of all protein pairs in set *train66k* and analyzed the five pairs (10 proteins) with the highest Euclidean distance in the same homologous superfamily (Supplementary Table S6). High distance within the same homologous superfamily indicates potentially wrong annotations. Secondly, we computed the nearest neighbors of those ten proteins to find an alternative, potentially more suitable annotation. For instance, the proteins in the *Phosphorylase Kinase* superfamily with the largest embedding distance (4pdyA01, bacterial aminoglycoside phosphotrans-ferase) to any other protein within this family (3skjF00, human *Galactose-binding domain-like* (77)) linked to different UniProt entries (*C8WS74_ALIAD* and *EPHA2_HUMAN*). In contrast, the nearest neighbor (3heiA00, human *phosphorylase kinase* (78)) of 3skjF00 linked to the same UniProt entry (*EPHA2_HUMAN* (68)) with the same enzymatic activity (EC number 2.7.10.1 (79)). Such analyses may indicate impure homologous superfamilies along with suggesting al-

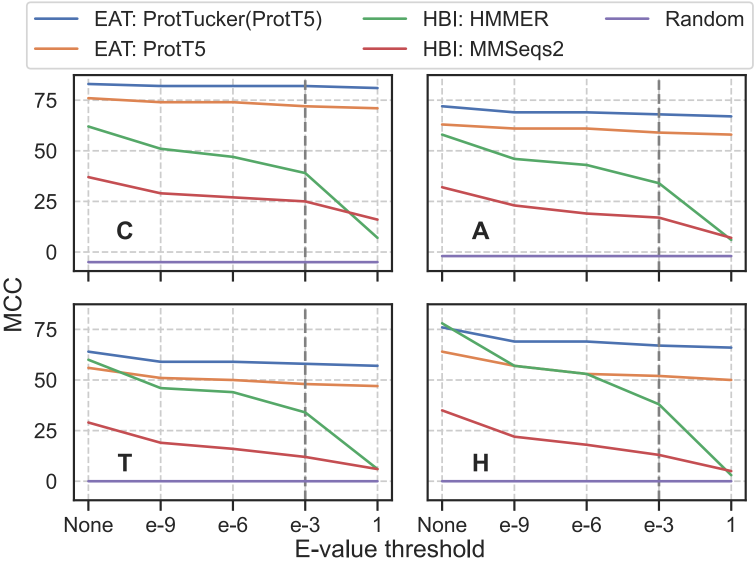

Figure 6. Performance decreasing with lower residual sequence similarity. We analyzed the change of performance in MCC (Equation 3) through removing proteins from *lookup69k* based on their *E*-value with respect to *test219* for two HBI-based (green: HMMER (74); red: MMSeqs2 (27)) and two EAT-based methods (orange: raw ProtT5 (37); blue: contrastive learning optimized ProtTucker(ProtT5)). The *E*-values were derived by searching sequences in *test219* against *lookup69k* using (i) HMM-profiles from CATH-Gene3D (54) through HMMer and (ii) MMSeqs2 sequence search with highest sensitivity (*-s 7.5, -cov 0*). 'None' referred to the performance without applying any threshold, i.e. all proteins in lookup69k were used for annotation transfer; all other thresholds referred to removing proteins below this *E*-value from *lookup69k*. Predictions were considered as false positives when no hit was found; pairs without CATH class matches were ignored. While the performance of EAT using raw ProtT5 and refined ProtTucker(ProtT5) embeddings decreased upon removing sequence similar pairs (toward right), HBI-based methods dropped significantly more. The default threshold for most sequence searches (*E*-value < 1e–3) was highlighted by vertical, gray, dashed lines.

ternative labels to be confirmed or rejected through manual curation.

EAT predicts entire proteomes in minutes

Training ProtTucker(ProtT5) required generating ProtT5 embeddings for *train66k*. This took 23m and 11m, respectively. Embeddings were generated using ProtT5 in half-precision with batch processing. All times were measured on a single Nvidia RTX A6000 with 48GB of vRAM and an AMD EPYC ROME 7352.

When predicting for new queries, ProtTucker requires *labeled lookup* proteins from which annotations can be transferred to unlabeled query proteins. Embeddings for this lookup set are pre-computed for the first query and can be re-used for all subsequent queries at any future time. The time required to labeled *lookup* proteins from which annotations can be transferred to unlabeled query proteins. Embeddings for this lookup set are pre-computed for the first query and can be re-used for all subsequent queries at any future time. The time required to generate ProtTucker embeddings from the embeddings of pLMs was negligible as its generation required only a single forward pass through a two-layer FNN. This implied that the total time for EAT with ProtTucker was largely determined by the embedding generation speed. For instance, creating per-protein embeddings from ProtT5 for the 123k proteins in CATH-S100 required 23 min (m). The total time for creating ProtTucker(ProtT5) embeddings from ProtT5 embeddings for the same set on the same machine was 0.5 seconds (s), i.e. ProtTucker added about 0.3%. Creating HMM profiles for the same set using either MSAs from MMSeqs2 (–*num-iterations 3, -s 7.5*) or jackhmmer took 15 m or 30 h, respectively (Table 3).

To predict using EAT, users have to embed only single query proteins requiring, on average, 0.01 s per protein for the CATH-S100 set. Using either single protein sequence search (MMSeqs2), pre-computed HMM profiles (HMMER) or pre-computed embeddings (ProtTucker) to transfer annotations from CATH-S100 to a single query protein took on average 0.025, 1.5 or 0.0008 s, respectively. Proteins

Table 3. Runtime*

Methods	Pre-processing (s)	Inference (s)
MMSeqs2 (sequence)	0.2×10^3	2.5×10^{-2}
HMMER (HMM)	114×10^3	150×10^{-2}
ProtTucker(ProtT5)	1.4×10^3	1.4×10^{-2}

* Runtime to transfer annotations from CATH-S100 (123k proteins) to a single query. All times measured in seconds [s] on a single Nvidia RTX A6000 with 48GB of vRAM and an AMD EPYC ROME. *Methods*: two HBI-based methods (MMSeqs2 and HMMER) and one EAT-based method (ProtTucker(ProtT5)). Pre-processing: measured the time required for building datasets (indexed database for MMSeqs2; MSA for jackhammer plus HMM profiles (HMMER) or ProtT5 embeddings (ProtTucker); Inference: the time for each new protein with a transfer.

in the PDB and CATH are, on average, roughly half as long (173 residues) as those from UniProt (343 residues). This is relevant for runtime, because embedding generation scales quadratically with sequence length (Figure 13 in SOM of (37)).

This increase was also reflected for the proteome-wide annotation transfer (Table 4), although these values included computations required for all aspects of EAT (1: load ProtT5 embeddings for pre-computed CATH-S100 lookup set; 2: load ProtT5 and embedding for query proteome; 3: generate ProtTucker(ProtT5) embeddings for queries and lookup; 4: compute pairwise Euclidean distances between query/lookup). We compared EAT using ProtTucker(ProtT5) embeddings to HBI proxied by existing Gene3D annotations taken from UniProt for three different proteomes (Table 4). At an expected error rate of 5% (Euclidean distance $\leq$ 0.9, Supplementary Table S3), EAT predicted substantially more proteins than Gene3D at HMMER *E*-value < 10^{-3}. For the subset of proteins for which both methods transferred annotations, those largely agreed (*Agreement*, Table 4; Supplementary Table S7 for other thresholds). All values for coverage decreased for multi-domain proteins, as proxied by 'multiple Gene3D annotations', while the agreement between Gene3D and ProtTucker(ProtT5) increased for two of three proteomes (Table 4: *multi*).

DISCUSSION

Prototype for representation learning of hierarchies

We have presented a new solution for combining the information implicitly contained in the embeddings from protein Language Models (pLMs) and contrastive learning to learn directly from hierarchically sorted data. As proof-of-concept, we applied the concept to the CATH hierarchy of protein structures (54,1,6,80). Hierarchies are difficult to handle by traditional supervised learning solutions. One *shortcut* is to learn each level in the hierarchy independently (81–83) at the price of having less information for other levels and of not explicitly benefiting from the hierarchy. Instead, our solution of contrastively learning protein triplets (anchor, positive, negative) to extract a new embedding space by condensing positives and moving negatives apart benefits from CATH's hierarchical structure. Simultaneously training a single, shared feed-forward neural network (FNN) on triplets from all four CATH classification levels allowed the network to directly capture the hierarchy. Encoding protein sequences through previously trained pLMs enabled ready information transfer from large but unlabeled sequence databases such as BFD (69) to 10,000-times smaller but experimentally annotated (labeled) proteins of known 3D structure classified by CATH. In turn, this allowed us to readily leverage aspects of protein structure captured by pLMs that are informative enough to predict structure from embeddings alone (52). Although the raw, pre-trained, unoptimized embeddings captured some aspects of the classification (Figures 2A–4A, Table 1), contrastive learning boosted this signal significantly (Figures 2B–4B, Table 1).

Crucial for this success was the novel combination of hierarchy- and batch-hard sampling (Table 2). Presumably, because those techniques enforce so-called *semi-hard* triplets that are neither too simple nor too hard to learn (72). This training setup learned different classifications for the same protein pair, depending on the third protein forming the triplet, thereby forcing the network to learn the complex hierarchy. The ambivalence in the notion of *positive/negative pair* facilitated training by allowing to include superfamilies with few members (otherwise to be skipped) and it increased the number of possible triplets manifold compared to only sampling on the level of superfamilies. These advantages might partially explain the synergy of both sampling techniques (Table 2).

Raw embedding EAT matched profile alignments in hit detection

In technical analogy to homology-based inference (HBI), we used embedding based annotation transfer (EAT, Figure 1B) to transfer annotations from labeled lookup proteins (proteins with a known CATH classification) to unlabeled query proteins (any protein of known sequence without known structure). Instead of transferring annotations from the closest hit in sequence space, EAT transferred annotations to the hit with smallest Euclidean distance in embedding space. This relatively simple approach was shown previously to predict protein function as defined by Gene Ontology (GO) better than hand-crafted features (50) even to levels competitive to much more complex approaches (48).

The concept of EAT was so successful that raw embeddings from two different pre-trained pLMs (ESM-1b (38), and ProtT5 (37)) already set the bar high for predicting CATH levels. The raw, general-purpose ESM-1b and ProtT5 outperformed HBI based on advanced HMM-profiles from HMMER (74) on the C- and A-level while falling short on the H-level (Table 1). Furthermore, we showed that ProtT5 already separated protein pairs with the same from those with different homologous superfamilies even when using a lookup set that consisted only of proteins with maximally 30% pairwise sequence identity (Figure 4A). Importantly, this competitive performance was achieved at a much smaller cost in terms of runtime (Tables 3 and 4).

As the lookup embeddings or HMM profiles are computed only once, we neglected this additional step. Such preparations cost much more than single queries: pre-computing HMM profiles using MMseqs2 took 15m, pre-computing embeddings about 23m (Table 3) using the same set and machine but utilizing CPUs in one (MMseqs2) and GPUs in the other (ProtT5). Only MMSeqs2 generated and indexed its database rapidly (19.5 s). However, pre-processing is required only once, rapidly amortizing when running many queries. The ability to pre-compute such representations is also a crucial difference between ProtTucker and other learned methods (44,59). For pairwise protein comparisons, those methods typically require N comparisons/forward-passes to search with a single query against N proteins. Instead, ProtTucker only needs a single forward pass to embed the new query; subsequent similarity scoring simply and quickly computes an Euclidean distance.

Table 4. CATH predictions for three model proteomes*

Proteome	Size	Gene3D	ProtTucker(ProtT5)@0.9	Agreement	Gene3D-multi	Agreement-multi	Inference time [s]
E. Coli (K12)	2033	59% (1193)	97% (1982)	81% (963)	23% (275)	86% (235)	113 (0.06)
A. Ostoyae	22 192	31% (6902)	79% (17 416)	75% (4707)	18% (1223)	65% (684)	1384 (0.06)
M. Chiliensis	1120	35% (392)	87% (974)	84% (320)	10% (40)	73% (29)	95 (0.09)

* Comparison of the annotation-transfer from 123k CATH-S100 proteins (5,54) through HBI (*Gene3D* (64)) and through EAT as introduced here (ProtTucker(ProtT5), or PT(ProtT5)) for three entire reference proteomes: *Escherichia coli* (*E. coli*), *Armillaria ostoyae* (*A. ostoyae*) and *Megavirus chilensis* (*M. chilensis*). In other words, all proteins in the three organisms were mapped to proteins of known structure using the CATH hierarchy. Gene3D predictions were taken from UniProt; PT(ProtT5) predictions were derived from the single nearest neighbor in Euclidean space. Coverage-related numbers refer to the percentage of proteins in the entire proteome (Size; in brackets: actual number of proteins), while those pertaining to the agreement are percentages of the set with annotations. *Size:* number of proteins; *Gene3D:* fraction of proteins with Gene3D annotation (coverage); *PT(ProT5)@0.9*: coverage of PT(ProtT5) at Euclidean distance < = 0.9 (5% error rate; Supplementary Table S3); *Agreement:* fraction of proteins for which Gene3D and PT(ProT5) had a prediction and reported the same homologous CATH superfamily (for multi-domain proteins with multiple Gene3D annotations, matching any domain by PT(ProtT5) was considered as correct); *Gene3D Multi:* fraction of Gene3D proteins with multi-domain annotation; *Agreement Multi:* fraction of multi-domain proteins for which the homologous CATH superfamily predicted by PT5 agreed with one of the Gene3D domain annotations; *Inference time:* the total time needed for proteome-wide embedding-based annotation transfer (EAT) measured in seconds [s] on a single Nvidia RTX A6000 with 48GB of vRAM and an AMD EPYC ROME (in brackets the average time per protein).

This makes ProtTucker search speed scale well with database growth suggesting the tool as a fast but sensitive pre-filter for other methods that in turn provide residue-level information as showcased on three model organisms (Table 4), including one of the largest organisms on earth (fungus *A. ostoyae*, 22 192 proteins) and one of the largest viruses (*M. chilensis*, 1120 proteins). In <27 min on a single machine (Table 4), ProtTucker transferred substantially more CATH annotations mapping proteins from their sequence to 3D structures through the CATH resource than Gene3D (64) at a similar level of expected error (Table 4).

For the virus and the bacterium (*E. coli*) the annotations agreed to over 80% with Gene3D, while this value dropped to 75% for the fungus (Table 4). Although high, the agreement was lower than expected: if ProtTucker and Gene3D each had fewer than 5% errors, then both should agree for over 90% of the proteins for which both transfer annotations. Most likely, this discrepancy (Δ(90-77)) arose partially from multi-domain proteins. Despite carefully cross-validating ProtTucker, an alternative explanation for the discrepancy is underestima-ting the expected error a distances $\leq$0.9 as 5% instead of up to 15%. The 'functional shape' of the agreement between ProtTucker and Gene3D at different distance thresholds (Supplementary Table S7) suggested that the 'errors' (lack of agreement) did not only originate from ProtTucker. Carefully annotating the five proteins with the lowest distance and a different CATH annotation (Supplementary Table S4) supported this perspective.

The agreement for multi-domain proteins dropped less than expected (11 percentage points drop for *M. Chilensis*, 5 percentage points increase for *E. coli*), possibly suggesting that ProtTucker using averages over an entire protein for comparison did not trip substantially more over the multi-domain challenge than the local alignment-based Gene3D using HMMER (74). This might suggest ProtTucker to have added correct annotations over Gene3D in multi-domain proteins, although developed exclusively on single domain proteins. The substantial increase in coverage from the level expected at distances $\geq$0.9 (Figure 5, Supplementary Table S4) for the proteomes (Table 4) might be misleading: to establish performance coverage (Figure 5, Supplementary Table S4), we used a highly non-redundant lookup set, presumably removing many easy hits. In contrast, analyzing proteomes, we transferred annotations for all CATH-S100 proteins, leveraging 'redundant annotation transfers' to increase coverage.

As for HBI, the accuracy of EAT also increased for larger families (Supplementary Figure S1). One explanation is that the larger the family, the higher the random hit rate, simply because there are more possible hits. Another, more subtle (and given the enormous compute time needed to train ProtT5, more difficult to test) explanation is that the largest CATH families represent most of the largest protein families (54). In fact, a few hundred of the largest superfamilies cover half of the entire sequence space (54,84). Simply due to their immense size, these large families have been sampled more during the pre-training of ProtT5.

ProtTucker embeddings intruded into midnight zone

The embedding space resulting from contrastive learning, introduced here, improved performance consistently for all four pLMs (Table 1). This was revealed through several ways of looking at the results from embeddings with and without contrastive learning: (i) the increased separation of protein pairs within the same protein superfamily and between different superfamilies (Figure 2), (ii) the qualitative improvement in the clustering of t-SNE projections (Figure 3), the better correlation of embedding distance and structural similarity (Figure 4) and (iii) the quantitative improvement in the EAT benchmark (Table 1). On top, the Euclidean distance correlated with accuracy (Figure 5, Supplementary Table S3). Similar to an E-value in HBI, this lets users gauge the reliability of a hit between query and annotated protein.

While the accuracy of the best performing pLM (ProtTucker(ProtT5)) was similar to HBI using HMM-profiles on the most fine-grained level of homologous superfamilies (CATH level H, Table 1), the relative advantage of EAT increased, the more diverged the level of inference, i.e. EAT outperformed HBI for more distant relations from the midnight zone (CATH level C, Table 1). When further reducing data redundancy, i.e. removing more similar sequences, this trend became clearer (Figure 6). Despite increasing difficulty, the performance of EAT decreased almost insignificantly where HBI approached random for insignificant E-values. This trend was supported by the correlation of structural similarity as defined by SSAP (65,66) and the Euclidean distance between protein pairs in a 30% non-redundant data set (Figure 4).

ProtTucker and tools such as HMMer have very different resolution: ProtTucker considers only per-protein averages to match query to template. In contrast, HMMer – or similar methods – align each residue between both proteins. The coarse-grain yields the speedup (Table 4), and pitches ProtTucker as a fast pre-filter. Once the hit is found by scanning large data sets, the slower, fine-grained methods for per-residue alignments and 3D prediction can be employed. However, the per-protein average also implies limitations, e.g. when Q and T have very different numbers of domains or the number of domains for Q is not known (Table 3).

Ultimately, the coarse-grained ProtTucker can compete at all because embeddings intrinsically abstract the constraints under which protein sequences evolve, including constraints upon structure, function, and the environment. The same constraints coin the evolutionary information contained in profiles of protein families. Apparently, pLMs such as ESM-1b (38), ProtBERT (70), or ProtT5 (70) are successfully condense these constraints. In fact, pLMs are arguably more successful than profile-based methods because a simple length-average over the position-specific scoring metrices (PSSM) would not suffice to predict CATH numbers very accurately.

ProtTucker builds upon this success to explicitly capture the constraints relevant for the CATH hierarchy. Thus, the less a particular aspect of function depends on structure, the less likely the new ProtTucker embeddings will reflect this aspect. On the other hand, an approach similar to ProtTucker focused on particular functional hierarchies, e.g. EC numbers appears to work well (SM Akmese & M Heinzinger, unpublished).

Taken together, these results indicated that contrastive learning captured structural hierarchies and provides a novel, powerful tool to uncover structural similarities clearly beyond what has been achievable with 50 years of optimizing sequence-based alignment techniques. Using EAT to complement HBI could become crucial for a variety of applications, ranging from finding remote structural templates for protein 3D structure predictions over prioritizing new proteins without any similarity to an existing structure to filtering potentially wrong annotations. One particular example has recently been shown for the proteome of SARS-CoV-2 to unravel entire functional components possibly relevant for fighting COVID-19 (61).

ProtTucker embeddings improved FunFams clustering

Previously (49), we showed that a simplistic predecessor of ProtTucker helped to refine the clustering of FunFams (76). By adding an additional, more fine-grained hierarchy level in CATH, FunFams link the structure-function continuum of proteins. The functional consistency within FunFams was proxied through the enzymatic activity as defined by the EC (Enzyme Commission (79)) number. Even the preliminary ProtTucker improved the annotation transfer of ligand binding and EC numbers (49) by removing outliers from existing FunFams and by creating new, more functionally coherent FunFams. As for CATH, the contrastively trained ProtTucker(ProtBERT) also improved over its unsupervised counterpart, ProtBERT, for FunFams. It improved functional consistency especially for proteins in the twilight zone (<35% PIDE, Figure 5 in (49)). Thus, ProtTucker embeddings improved functional (FunFams) and structural (CATH) consistency beyond sequence similarity. Here, we expanded upon this analysis by showing how EAT can be improved even more through contrastively learning hierarchies. Using the proposed method, we could spot potential outliers, i.e. samples with the same annotation but large embedding distance. This might become essential to clean up databases. Aside from outlier-spotting, we could also obtain labels from the nearest neighbors of outliers (Supplementary Table S6). Although we could not reproduce the same level of success when applying EAT to inferring subcellular location in ten states (Table 3), the CATH-optimized ProtTucker embeddings also did not perform worse.

Generic advantages of *contrastive learning*

Contrastive learning benefits from hierarchies as opposed to supervised training which usually flattens the hierarchy thereby losing its intrinsic advantage. Other possible advantages of contrastive learning include the following three. (i) Dynamic data update (*online learning*): While supervised networks require re-training to benefit from new data, contrastively trained networks can benefit from new data by simply updating the lookup set. This could even add completely new classes, such as proteins for which the classification will become available only in the future. HBI shares this advantage that originates from the difference between classifying proteins into existing families versus classifying by identifying the most similar proteins in that family. (ii) Learn the access, not the data: Instead of forcing the supervised network to memorize the training data, contrastive learning teaches how to access the data stored in an external lookup set. (iii) Compression: As many other learning techniques, contrastive learning can act as a compression technique. For instance, we reduced the disk space required to store protein embeddings threefold by projecting 1024-dimensional vectors onto 128 dimensions while improving performance (Table 1). This renders new queries (inference) more efficient and enables scaling up to very large lookup sets. (iv) Interpretability: Knowing from which protein an annotation was transferred might help users benefit more from a certain prediction than just the prediction itself. For instance, knowing that an unnamed query protein shares all CATH levels with a particular glucocorticoid receptor might suggest some functional implications helping to design future experiments.

CONCLUSIONS

Embeddings from protein Language Models (pLMs) extract the information learned by these models from unlabeled protein sequences. Embedding-based Annotation

Transfer (EAT) replacing the proximity in sequence space used by homolog-based inference (HBI) through proximity in embedding space already reaches traditional alignment methods in transferring CATH annotations from a template protein with experimental annotations to an unlabeled query protein. Although not quite reaching the performance of advanced profile-profile searches by HMMer for all four CATH levels, the best embeddings surpassed HMMer for two of the four levels (C and A). When optimizing embeddings through contrastive learning for the goal of transferring CATH annotations, EAT using these new embeddings consistently outperformed all sequence comparison techniques tested. This higher performance was reached at a fraction (three orders of magnitude) of the computational time. Although the new embeddings optimized through contrastive learning for CATH did not improve performance for a completely different task, namely the prediction of subcellular location in ten classes, the CATH-optimized solution did also not perform significantly worse. Remarkably, just like HBI, the performance of EAT using the optimized ProtTucker embeddings was proportional to family size with increased accuracy for larger families.

DATA AVAILABILITY

Building on top of bio_embeddings package (85) we have made a script available that simplifies EAT https://github.com/Rostlab/EAT.

SUPPLEMENTARY DATA

Supplementary Data are available at NARGAB Online.

ACKNOWLEDGEMENTS

Thanks primarily to Tim Karl for invaluable help with hardware and software; to Inga Weise (TUM) for support with many other aspects of this work. Many thanks for both anonymous reviewers, deep kudos for the detailed help from No. 1! Last, but not least, thanks to all those who maintain public databases in particular Steven Burley (PDB, Rutgers), Alan Bridge (Swiss-Prot, SIB, Lausanne), Alex Bateman (UniProt, EBI Hinxton) and their crews, and to all experimentalists who enabled this analysis by making their data publicly available.

FUNDING

Bavarian Ministry of Education through funding to the TUM and by a grant from the Alexander von Humboldt foundation through the German Ministry for Research and Education (BMBF: Bundesministerium für Bildung und Forschung); BMBF [031L0168 and program 'Software Campus 2.0 (TUM) 2.0' 01IS17049];Deutsche Forschungsgemeinschaft [DFG-GZ: RO1320/4-1].

Conflict of interest statement. None declared.

REFERENCES

1. Das,S., Sillitoe,I., Lee,D., Lees,J.G., Dawson,N.L., Ward,J. and Orengo,C.A. (2015) CATH funfhmmer web server: protein functional annotations using functional family assignments. *Nucleic Acids Res.*, **43**, W148–W153.
2. Sonnhammer,E.L. and Kahn,D. (1994) Modular arrangement of proteins as inferred from analysis of homology. *Protein Sci.*, **3**, 482–492.
3. Bateman,A., Birney,E., Durbin,R., Eddy,S.R., Finn,R.D. and Sonnhammer,E.L. (1999) Pfam 3.1: 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucleic Acids Res.*, **27**, 260–262.
4. Gough,J. and Chothia,C. (2002) SUPERFAMILY: HMMs representing all proteins of known structure. SCOP sequence searches, alignments and genome assignments. *Nucleic Acids Res.*, **30**, 268–272.
5. Orengo,C.A., Flores,T.P., Taylor,W.R. and Thornton,J.M. (1993) Identification and classification of protein fold families. *Protein Eng.*, **6**, 485–500.
6. Orengo,C.A., Michie,A.D., Jones,S., Jones,D.T., Swindells,M.B. and Thornton,J.M. (1997) CATH - a hierarchic classification of protein domain structures. *Structures*, **5**, 1093–1108.
7. Todd,A.E., Orengo,C.A. and Thornton,J.M. (2001) Evolution of function in protein superfamilies, from a structural perspective. *J. Mol. Biol.*, **307**, 1113–1143.
8. Yona,G. and Levitt,M. (2002) Within the twilight zone: a sensitive profile-profile comparison tool based on information theory. *J. Mol. Biol.*, **315**, 1257–1275.
9. Doolittle,R.F., Feng,D.-F., Johnson,M.S. and McClure,M.A. (1986) Origins and evolutionary relationships of retroviruses. *Q. Rev. Biol.*, **64**, 1–30.
10. Rost,B. (1999) Twilight zone of protein sequence alignments. *Protein. Eng.*, **12**, 85–94.
11. Rost,B. (1997) Protein structures sustain evolutionary drift. *Fold. Des.*, **2**, S19–S24.
12. Mika,S. and Rost,B. (2003) UniqueProt: creating representative protein sequence sets. *Nucleic Acids Res.*, **31**, 3789–3791.
13. Rost,B. (2002) Enzyme function less conserved than anticipated. *J. Mol. Biol.*, **318**, 595–608.
14. Nehrt,N.L., Clark,W.T., Radivojac,P. and Hahn,M.W. (2011) Testing the ortholog conjecture with comparative functional genomic data from mammals. *PLoS Comput. Biol.*, **7**, e1002073.
15. Sander,C. and Schneider,R. (1991) Database of homology-derived structures and the structural meaning of sequence alignment. *Proteins*, **9**, 56–68.
16. Higgins,D.G., Bleasby,A.J. and Fuchs,R. (1992) CLUSTAL V: improved sofware for multiple sequence alignment. *CABIOS*, **8**, 189–191.
17. Thompson,J., Higgins,D. and Gibson,T. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4690.
18. Sjölander,K., Karplus,K., Brown,M.P., Hughey,R., Krogh,A., Mian,I.S. and Haussler,D. (1996) Dirichlet mixtures: a method for improving detection of weak but significant protein sequence homology. *CABIOS*, **12**, 327–345.
19. Altschul,S.F., Madden,T.L., Schaeffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped blast and PSI-Blast: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
20. Eddy,S.R. (1998) Profile hidden markov models. *Bioinformatics*, **14**, 755–763.
21. Jaroszewski,L., Rychlewski,L. and Godzik,A. (2000) Improving the quality of twilight-zone alignments. *Protein Sci.*, **9**, 1487–1496.
22. Sadreyev,R. and Grishin,N. (2003) COMPASS: a tool for comparison of multiple protein alignments with assessment of statistical significance. *J. Mol. Biol.*, **326**, 317–336.
23. Edgar,R.C. and Sjolander,K. (2004) COACH: profile-profile alignment of protein families using hidden markov models. *Bioinformatics*, **20**, 1309–1318.
24. Wang,G. and Dunbrack,R.L. Jr (2004) Scoring profile-to-profile sequence alignments. *Protein Sci.*, **13**, 1612–1626.
25. Soding,J. (2005) Protein homology detection by HMM-HMM comparison. *Bioinformatics*, **21**, 951–960.
26. Sievers,F., Wilm,A., Dineen,D., Gibson,T.J., Karplus,K., Li,W., Lopez,R., McWilliam,H., Remmert,M., Soding,J. *et al.* (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Mol. Syst. Biol.*, **7**, 539.

27. Steinegger,M. and Soding,J. (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, **35**, 1026–1028.
28. Przybylski,D. and Rost,B. (2007) Consensus sequences improve PSI-BLAST through mimicking profile-profile alignments. *Nucleic Acids Res.*, **35**, 2238–2246.
29. Rost,B., Liu,J., Nair,R., Wrzeszczynski,K.O. and Ofran,Y. (2003) Automatic prediction of protein function. *Cell. Mol. Life Sci.*, **60**, 2637–2650.
30. Rost,B. (1996) PHD: predicting one-dimensional protein structure by profile based neural networks. *Meth Enzymol*, **266**, 525–539.
31. Rost,B. and Sander,C. (1993) Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.*, **232**, 584–599.
32. Jumper,J., Evans,R., Pritzel,A., Green,T., Figurnov,M., Ronneberger,O., Tunyasuvunakool,K., Bates,R., Zidek,A., Potapenko,A. *et al.* (2021) Highly accurate protein structure prediction with alphafold. *Nature*, **569**, 583–589.
33. Baek,M., Dimaio,F., Anishchenko,I., Dauparas,J., Ovchinnikov,S., Lee,G.R., Wang,J., Cong,Q., Kinch,L.N., Schaeffer,R.D. *et al.* (2021) Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, **373**, 871–876.
34. Peters,M.E., Neumann,M., Iyyer,M., Gardner,M., Clark,C., Lee,K. and Zettlemoyer,L. (2018) Deep contextualized word representations. arXiv doi: https://doi.org/10.48550/arXiv.1802.05365, 22 March 2018, preprint: not peer reviewed.
35. Devlin,J., Chang,M.-W., Lee,K. and Toutanova,K. (2019) *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, pp. 4171–4186.
36. Brown,T.B., Mann,B., Ryder,N., Subbiah,M., Kaplan,J., Dhariwal,P., Neelakantan,A., Shyam,P., Sastry,G., Askell,A. *et al.* (2020) Language models are few-shot learners. arXiv doi: https://doi.org/10.48550/arXiv.2005.14165, 22 July 2020, preprint: not peer reviewed.
37. Elnaggar,A., Heinzinger,M., Dallago,C., Rehawi,G., Wang,Y., Jones,L., Gibbs,T., Feher,T., Angerer,C., Steinegger,M. *et al.* (2021) ProtTrans: towards cracking the language of life's code through self-supervised learning. *IEEE TPAMI*, **14**, 30.
38. Rives,A., Meier,J., Sercu,T., Goyal,S., Lin,Z., Liu,J., Guo,D., Ott,M., Zitnick,C.L., Ma,J. *et al.* (2021) Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl Acad. Sci. U.S.A.*, **118**, e2016239118.
39. Alley,E.C., Khimulya,G., Biswas,S., AlQuraishi,M. and Church,G.M. (2019) Unified rational protein engineering with sequence-based deep representation learning. *Nature Meth*, **16**, 1315–1322.
40. Heinzinger,M., Elnaggar,A., Wang,Y., Dallago,C., Nechaev,D., Matthes,F. and Rost,B. (2019) Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinf.*, **20**, 723.
41. Rao,R., Meier,J., Sercu,T., Ovchinnikov,S. and Rives,A. (2020) Transformer protein language models are unsupervised structure learners. bioRxiv doi: https://doi.org/10.1101/2020.12.15.422761, 15 December 2020, preprint: not peer reviewed.
42. Madani,A., McCann,B., Naik,N., Shirish Keskar,N., Anand,N., Eguchi,R.R., Huang,P. and Socher,R. (2020) ProGen: language modeling for protein generation. arXiv doi: https://doi.org/10.48550/arXiv.2004.03497, 8 March 2020, preprint: not peer reviewed.
43. Ofer,D., Brandes,N. and Linial,M. (2021) The language of proteins: NLP, machine learning & protein sequences. *Comp Structural Biotechn J*, **19**, 1750–1758.
44. Bepler,T. and Berger,B. (2021) Learning the protein language: evolution, structure, and function. *Cell Syst.*, **12**, 654–669.
45. Bepler,T. and Berger,B. (2019) Learning protein sequence embeddings using information from structure. *Seventh International Conference on Learning Representations*.
46. Stärk,H., Dallago,C., Heinzinger,M. and Rost,B. (2021) Light attention predicts protein location from the language of life. *Bioinformatics Adv.*, **1**, vbab035..
47. Littmann,M., Heinzinger,M., Dallago,C., Weissenow,K. and Rost,B. (2021) Protein embeddings and deep learning predict binding residues for various ligand classes. *Sci. Rep.*, **11**, 23916.
48. Littmann,M., Heinzinger,M., Dallago,C., Olenyi,T. and Rost,B. (2021) Embeddings from deep learning transfer GO annotations beyond homology. *Sci. Rep.*, **11**, 1160.
49. Littmann,M., Bordin,N., Heinzinger,M., Schütze,K., Dallago,C., Orengo,C. and Rost,B. (2021) Clustering funfams using sequence embeddings improves EC purity. *Bioinformatics*, **37**, 3449–3455.
50. Villegas-Morcillo,A., Makrodimitris,S., van Ham,R.C.H.J., Gomez,A.M., Sanchez,V. and Reinders,M.J.T. (2021) Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, **37**, 162–170.
51. Hamid,M.-N. and Friedberg,I. (2019) Identifying antimicrobial peptides using word embedding with deep recurrent neural networks. *Bioinformatics*, **35**, 2009–2016.
52. Weißenow,K., Heinzinger,M. and Rost,B. (2022) Protein language model embeddings for fast, accurate, alignment-free protein structure prediction. *Structure*, https://doi.org/10.1016/j.str.2022.05.001.
53. Le-Khac,P.H., Healy,G. and Smeaton,A.F. (2020) *Contrastive Representation Learning: A Framework and Review*. IEEE Access.
54. Sillitoe,I., Bordin,N., Dawson,N., Waman,V.P., Ashford,P., Scholes,H.M., Pang,C.S.M., Woodridge,L., Rauer,C., Sen,N. *et al.* (2021) CATH: increased structural coverage of functional space. *Nucleic Acids Res.*, **49**, D266–D273.
55. Fox,N.K., Brenner,S.E. and Chandonia,J.-M. (2014) SCOPe: structural classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Res.*, **42**, D304–D309.
56. Nallapareddy,V., Bordin,N., Sillitoe,I., Heinzinger,M., Littmann,M., Waman,V., Sen,N., Rost,B. and Orengo,C. (2022) CATHe: detection of remote homologues for CATH superfamilies using embeddings from protein language models. bioRxiv doi: https://doi.org/10.1101/2022.03.10.483805,13 March 2022, preprint: not peer reviewed.
57. Li,C.-C. and Liu,B. (2019) MotifCNN-fold: protein fold recognition based on fold-specific features extracted by motif-based convolutional neural networks. *Brief Bioinform*, **21**, 2133–2141.
58. Liu,B., Li,C.-C. and Yan,K. (2020) DeepSVM-fold: protein fold recognition by combining support vector machines and pairwise sequence similarity scores generated by deep learning networks. *Brief Bioinform*, **21**, 1733–1741.
59. Gao,M. and Skolnick,J. (2021) A novel sequence alignment algorithm based on deep learning of the protein folding code. *Bioinformatics*, **37**, 490–496.
60. Chen,J., Guo,M., Wang,X. and Liu,B. (2018) A comprehensive review and comparison of different computational methods for protein remote homology detection. *Brief Bioinform*, **19**, 231–244.
61. O'Donoghue,S.I., Schafferhans,A., Sikta,N., Stolte,C., Kaur,S., Ho,B.K., Anderson,S., Procter,J., Dallago,C., Bordin,N. *et al.* (2021) SARS-CoV-2 structural coverage map reveals viral protein assembly, mimicry, and hijacking mechanisms. *Mol. Syst. Biol.*, **12**, e10079.
62. Burley,S.K., Berman,H.M., Bhikadiya,C., Bi,C., Chen,L., Di Costanzo,L., Christie,C., Dalenberg,K., Duarte,J.M., Dutta,S. *et al.* (2019) RCSB protein data bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic Acids Res.*, **47**, D464–D474.
63. Chen,T., Kornblith,S., Norouzi,M. and Hinton,G. (2020) *International Conference on Machine Learning*. PMLR, pp. 1597–1607.
64. Lewis,T.E., Sillitoe,I., Dawson,N., Lam,S.D., Clarke,T., Lee,D., Orengo,C. and Lees,J. (2018) Gene3D: extensive prediction of globular domains in proteins. *Nucleic Acids Res.*, **46**, D435–D439.
65. Taylor,W.R. and Orengo,C.A. (1989) A holistic approach to protein structure alignment. *Protein. Eng.*, **2**, 505–519.
66. Orengo,C.A. and Taylor,W.R. (1996) SSAP: sequential structure alignment program for protein structure comparison. *Meth Enzymol*, **266**, 617–635.
67. Almagro Armenteros,J.J., Sonderby,C.K., Sonderby,S.K., Nielsen,H. and Winther,O. (2017) DeepLoc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, **33**, 3387–3395.
68. The UniProt Consortium. (2021) UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.*, **49**, D480–D489.

69. Steinegger,M., Mirdita,M. and Soding,J. (2019) Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods*, **16**, 603–606.
70. Raffel,C., Shazeer,N., Roberts,A., Lee,K., Narang,S., Matena,M., Zhou,Y., Li,W. and Liu,P.J. (2020) Exploring the limits of transfer learning with a unified Text-to-Text transformer. *J Mach Learning Res*, **21**, 1–67.
71. Marquet,C., Heinzinger,M., Olenyi,T., Dallago,C., Erckert,K., Bernhofer,M., Nechaev,D. and Rost,B. (2021) Embeddings from protein language models predict conservation and variant effects. *Hum. Genet.*, https://doi.org/10.21203/rs.3.rs-584804/v1.
72. Hermans,A., Beyer,L. and Leibe,B. (2017) In defense of the triplet loss for person re-identification. arXiv doi: https://doi.org/10.48550/arXiv.1703.07737, 21 November 2017, preprint: not peer reviewed.
73. Kingma,D.P. and Ba,J. (2015) Adam: a method for stochastic optimization. arXiv doi: https://doi.org/10.48550/arXiv.1412.6980, 30 January 2017, preprint: not peer reviewed.
74. Finn,R.D., Clements,J. and Eddy,S.R. (2011) HMMER web server: interactive sequence similarity searching. *Nucleic Acids Res.*, **39**, W29–W37.
75. Pedregosa,F., Varoquaux,G., Gramfort,A., Michel,V., Thirion,B., Grisel,O., Blondel,M., Prettenhofer,P., Weiss,R. and Dubourg,V. (2011) Scikit-learn: machine learning in python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
76. Sillitoe,I., Cuff,A.L., Dessailly,B.H., Dawson,N.L., Furnham,N., Lee,D., Lees,J.G., Lewis,T.E., Studer,R.A., Rentzsch,R. *et al.* (2013) New functional families (FunFams) in CATH to improve the mapping of conserved functional sites to 3D structures. *Nucleic Acids Res.*, **41**, D490–D498.
77. Peng,L., Oganesyan,V., Damschroder,M.M., Wu,H. and Dall'Acqua,W.F. (2011) Structural and functional characterization of an agonistic anti-human epha2 monoclonal antibody. *J. Mol. Biol.*, **413**, 390–405.
78. Himanen,J.P., Goldgur,Y., Miao,H., Myshkin,E., Guo,H., Buck,M., Nguyen,M., Rajashankar,K.R., Wang,B. and Nikolov,D.B. (2009) Ligand recognition by A-class eph receptors: crystal structures of the epha2 ligand-binding domain and the epha2/ephrin-A1 complex. *EMBO Rep.*, **10**, 722–728.
79. Webb,E.C. (1992) *Enzyme Nomenclature 1992. Recommendations of the Nomenclature committee of the International Union of Biochemistry and Molecular Biology*. 1992 edn. Academic Press, New York.
80. Sillitoe,I., Dawson,N., Lewis,T.E., Das,S., Lees,J.G., Ashford,P., Tolulope,A., Scholes,H.M., Senatorov,I., Bujan,A. *et al.* (2019) CATH: expanding the horizons of structure-based functional annotations for genome sequences. *Nucleic Acids Res.*, **47**, D280–D284.
81. Jensen,L.J., Gupta,R., Blom,N., Devos,D., Tamames,J., Kesmir,C., Nielsen,H., Staerfeldt,H.H., Rapacki,K., Workman,C. *et al.* (2002) Prediction of human protein function from post-translational modifications and localization features. *J. Mol. Biol.*, **319**, 1257–1265.
82. Nair,R. and Rost,B. (2005) Mimicking cellular sorting improves prediction of subcellular localization. *J. Mol. Biol.*, **348**, 85–100.
83. Kernytsky,A. and Rost,B. (2009) Using genetic algorithms to select most predictive protein features. *Proteins*, **75**, 75–88.
84. Dessailly,B.H., Nair,R., Jaroszewski,L., Fajardo,J.E., Kouranov,A., Lee,D., Fiser,A., Godzik,A., Rost,B. and Orengo,C. (2009) PSI-2: structural genomics to cover protein domain family space. *Structure*, **17**, 869–881.
85. Dallago,C., Schuetze,K., Heinzinger,M., Olenyi,T., Littmann,M., Lu,A.X., Yang,K.K., Min,S., Yoon,S., Morton,J.T. *et al.* (2021) Learned embeddings from deep learning to visualize and predict protein sets. *Curr. Protoc.*, **1**, e113.
86. Van der Maaten,L. and Hinton,G. (2008) Visualizing data using t-SNE. *J. Mach. Learn. Res.*, **9**, 2579–2605.

6. Conclusion

Proteins play a pivotal role for correct cell functioning in any living organism. The smallest deviation in the order of their constituent parts, amino acid, might prevent a protein from performing its allocated function(s) with potentially lethal consequences for their host cell or the whole organism [115]. This makes understanding a protein's role within the carefully orchestrated and highly intertwined cellular machinery an interesting research topic for various applications in, e.g., precision medicine. Additionally, the life science sector in general and biotechnology in particular depends on an in-depth understanding of a protein's function to leverage proteins for highly specialized applications. This trend gets fueled by a rapid increase in high-throughput sequencing capabilities. While this led to a steady growth of publicly available protein sequence databases, it also increases the share of proteins for which we do not know anything but their sequence. Today, we have billions of protein sequences available that are only one mouse click away but only a tiny fraction thereof has additional (experimental) information on function or structure.

One of the most important challenges for computational biology is to bridge this gap between large but unlabeled and small but labeled data by computational means [5]. This raises two questions: a) Can we somehow leverage the vast amount of unlabeled protein sequence data, and b) how do we best represent protein sequences to make them machine-readable. The latter point refers to how to make proteins "machine-understandable" in a sense that they do not only hold information on the plain order of amino acids but also harbor already some information on their structure or function. For almost three decades, most state-of-the-art approaches tackled both questions by combining machine learning (ML) and explicit evolutionary information (EI) condensed in multiple sequence alignments (MSA) [13, 15, 66]. Even the highly accurate 3D structure prediction of AlphaFold 2 (AF2) [3] relies on exactly this combination with the crucial difference that information extraction from the MSAs is part of the training. While those approaches improved over time by employing more sophisticated ML algorithms and the way they leveraged the sequences in the MSA [3, 105], some major limitations inherent to EI remained. For example, for some applications, such as processing of large-scale metagenomic sets such as BFD [6, 7], retrieving related proteins from exponentially growing sequence databases is becoming too time-consuming or will require high-performance computing (HPC) infrastructure. But even if compute is no limiting factor, EI will remain less informative not only for small families, e.g., for proteins from the Dark Proteome [106], but also for applications that require protein-specific instead of family-averaged predictions, e.g., single amino acid variant (SAV) effect prediction.

Representation Learning for Proteins. In this dissertation, I proposed an alternative way to represent single protein sequences which leverages large but unlabeled protein sequence databases via representation learning. In contrast to previous approaches which relied on handcrafted features or features derived by statistical means, the proposed approaches learn to represent data directly from raw data itself, i.e., without any human intervention and without the need for any annotation but the sequence itself. Broadly speaking, this was achieved by adapting so called language models (LMs) from natural language processing (NLP) to protein sequences by equating protein sequences with sentences and words with amino acids. The sequential nature of both data formats allowed us to build on existing optimization concepts such as auto-regressive language modeling (ALM), which tries to predict the next amino acid/word given all previous amino acids/words in the same sequence/sentence. When being applied to millions or billions of protein sequences, this strategy forces the protein language model (pLM) to learn certain commonalities, patterns, or features directly from raw data. The acquired knowledge can later be transferred to other tasks by extracting the hidden states of the (p)LM upon receiving an input sequence (transfer learning).

In a first proof-of-principle, we introduced a novel way to represent single protein sequences as continuous vectors (embeddings) by using the LSTM-based [116] LM ELMo [79] taken from NLP. By modeling protein sequences instead of natural language, ELMo effectively captured the biophysical properties of the language of life as defined by the grammar of protein sequences from unlabeled big data (UniRef50 [82]). This provides a single-sequence based alternative over EI to make protein sequence machine-readable. Dropping the computationally heavy search for related sequences provides a significant speed-up: Where the highly sensitive HHblits [53] needed on average about two minutes to gather EI for a target protein, SeqVec created embeddings on average in 0.03s per protein. We refer to these new embeddings as *SeqVec* (Sequence-to-Vector; Chapter 2 [81]) and assess their effectiveness using predictive power as acid test. For example, we used SeqVec embeddings as input to a ML device in order to predict secondary structure and disorder. While the proposed solution outperformed other single-sequence-based representations, e.g., one-hot encoding or ProtVec [72], it fell short compared to established EI-based methods. This trend got confirmed when predicting subcellular localization and when distinguishing membrane-bound from water-soluble proteins.

In a second step, we used SeqVec embeddings (Chapter 2) to develop a new and simple method, dubbed *goPredSim* [37], to predict the function of a protein in the form of Gene Ontology (GO) [108] terms from single protein sequences (Chapter 3 - goPredSim). The proposed solution works similar to homology-based inference (HBI) which transfers annotations from a set of labeled proteins to a set of unlabeled proteins. However, instead of relying on sequence similarity, we used Euclidean distance between SeqVec embeddings to define similarity. To put the performance of this general-purpose solution which we refer to as *embedding-based annotation transfer* (EAT) into perspective of other methods, we replicated the conditions of the third Critical Assessment of protein Function Annotation algorithms (CAFA3) [38]. According to this benchmark and a preliminary evaluation of CAFA4 at ISMB2020 [109], goPredSim is competitive to top methods that are hand-tailored for GO term prediction. That our extremely simple and generally applicable approach remained competitive to sophisticated ML approaches, highlights the effectiveness of our proposed protein sequence representations. On top, in direct comparison, EAT clearly outperformed HBI, indicating that embeddings capture functional similarity beyond sequence similarity.

While our proof-of-principle, SeqVec, was based on a relatively small and shallow, LSTM-based LM, we later improved over our previous work by training two auto-regressive Transformers (Transformer-XL [94], XLNet [95]) and four auto-encoder Transformers (BERT [2], Albert [91], Electra [92], T5 [93]) on data from two large but unlabeled protein sequence databases containing up to 393 billion amino acids [28]. However, this up-scaling is not for free: Training such large LMs on huge amounts of data requires HPC infrastructure. The crucial point, however, is that the expensive pre-training has to be performed only once while any later deployment is extremely fast and requires only a desktop machine with a gaming GPU. Additionally, by relying only on single protein sequences to generate embeddings, we do not only bypass the expensive database searches but also the resulting family-average of MSAs/EI which enables highly accurate, protein-specific predictions. Taken together, our results implied that pLMs learned some of the grammar of the language of life.

As we were now equipped with highly informative, protein-specific ProtTrans embeddings, we developed *variant effect score prediction without alignments* (VESPA - Chapter 4 [103]). In a first step, we showed that ProtT5 embeddings alone predicted residue conservation almost as accurately from single sequences as MSA-based methods. In a second step, we used the conservation prediction along with BLOSUM62 [50] substitution scores and pLM mask reconstruction probabilities as input to a simplistic logistic regression ensemble to distinguish effect from neutral mutations. Comparing predictions for a standard set of 39 Deep Mutational Scanning experiments to other methods (incl. ESM-1v [112], DeepSequence [113], and GEMME [114]) revealed our approach as competitive with the state-of-the-art methods using MSA input at a fraction of the costs in computing/energy.

In a last step, we used contrastive learning to improve over ProtTrans embeddings by learning a new set of task-specific embeddings. Towards this end, a new set of embeddings was learnt that optimizes constraints captured by the hierarchical classifications of protein 3D structures defined by the CATH [17] resource. More specifically, we optimized the new embedding space towards pushing apart proteins with dissimilar CATH annotation while pulling together those proteins with similar annotations. After this training, EAT is again used to make predictions as no direct class output but a new embedding space is learnt. The approach, dubbed ProtTucker (Chapter 5 [104]), was shown to have an improved ability to recognize distant homologous relationships compared to more traditional techniques such as threading or fold recognition. These allowed the new embeddings to step into the “midnight zone” [56] of protein similarity, i.e., the region in which distantly related sequences have a seemingly random pairwise sequence similarity. The novelty of this work is the particular combination of tools and sampling techniques that ascertained good performance comparable or better to existing state-of-the-art sequence comparison methods.

Advantages of operating on single sequences. In summary, this dissertation showed that learning protein sequence representation directly from large, unlabeled data can enable single-sequence predictions at the accuracy of sophisticated MSA-based solutions. This improves speed to an extent where highly accurate predictions can be made for whole proteomes in less than an hour. Such predictions offer a perfect pre-filter for quickly sifting through large amounts of data before investigating the most promising candidates in more detail, e.g., via AF2. Protein-specific instead of family-averaged predictions offer an alternative use-case of the proposed embeddings. This will become increasingly important when trying to understand the subtle effect of single changes in a protein's amino acid arrangement as shown in Chapter 4. Additionally, we proposed an alternative to traditional HBI which relies on distance in embedding space rather than sequence-similarity for annotation transfer (Chapter 3 and Chapter 5). We could show in [117] that this single-sequence based approach can improve family-averaged clustering of functional families (FunFams, [20]) within CATH.

Do pLMs capture evolutionary information? As pLMs reached the performance of EI-based methods, one question that naturally arises is whether pLMs captured EI to a certain extent. Despite being straightforward to formulate, it is difficult to probe this question as EI refers to a rather abstract concept that cannot be precisely defined in e.g. mathematical terms. While the first step towards gathering EI, i.e., summarizing evolutionary related sequences that presumably share the same function and structure as MSA, is clear, there are multiple ways to actually extract information from it, e.g., *inter alia* PSSMs, HMMS, ECs. By now, even pLMs trained directly on MSAs learn to extract information [105]. However, there is no common standard that would allow to directly probe whether pLMs capture EI. A good approximation towards answering this question might be that embedding space reflects functional (Chapter 3) and structural similarity (Chapter 5) beyond plain sequence similarity.

When considering how those pLMs are trained, i.e., via ALM or MLM, one could also argue that pLMs capture EI to a certain extent as they can only do well in those tasks when learning certain commonalities between different, yet related sequences. This starts with local, reoccurring patterns such as certain secondary structure elements or binding motifs that are formed by a diverse spectrum of amino acid stretches. This is similar to synonyms of words that we observe in many languages, e.g., liberty and freedom refer to nearly identical ideas and hence result in similar embeddings. Similarly, different sequences can adopt the same structure and perform the same function, i.e., sequence-space is much larger than structure-space, as highlighted by the fact that there are considerably less protein folds compared to the large pool of sequences they were derived of [18]. While such information is never provided, neither in NLP nor in CB, (p)LMs are able to detect such commonalities if trained on enough data. In fact, the only way to perform well in optimization tasks such as ALM and MLM is to detect such patterns. Consequently, it is reasonable to assume that pLMs capture EI to a certain extent but it remains difficult to quantify to which extent. Still, the main advantage of pLMs over EI-based methods remains: While the former capture this information indirectly during pre-training and can transfer this knowledge to novel single sequences, the latter need to gather it explicitly from large databases for each new query.

What might be next? - Representation learning and ML in general are developing at such breathtaking speed that it is difficult to provide an all-embracing, long-term prediction. Therefore, I will rather focus on the short- to mid-term developments: Only recently, it was shown that representations learnt from MSAs [105] instead of single protein sequence improved structure prediction significantly [3]. This shows that there is still room for improvement of single-sequence based approaches as a protein sequence contains all the information required for protein folding [9]. Towards this end, more efficient algorithms or optimization strategies might be needed as our results [28] indicated that our models would still improve from seeing more samples despite being trained on HPC infrastructure. This is in line with recent conclusions suggesting that most of today's large LMs are underfitting [118]. An alternative strategy would be to continue the path of end-to-end learning by making even the database search part of training. Recent results showed that this is possible, despite only for relatively small databases of a few hundred thousand entries [119]. Another direction moves away from sequences and instead focuses on the millions of highly accurate protein 3D structure predictions from AF2 that should be made publicly available thanks to a collaboration between DeepMind and EMBL-EBI [120]. Suddenly, sequences will not be the only data modality available at high quality and large scale for proteins, opening the question on how to best leverage this plethora of 3D structures. First approaches were already presented that predict a protein sequence from its structure [121]. While the output of this task is interesting for protein engineering, it also provides embeddings for protein structures that can be used to make protein structures readable for any other task trained on top of such embeddings. Presumably, those two developments, improving single sequence representations and learning directly from protein 3D structures, will be combined in a unified, multimodal framework. The resulting embeddings would entangle information from sequence and structure space, rendering the resulting embeddings highly informative to predict functional aspects of proteins. A similar development can already be observed today in the field of CV where image and text data are embedded into a single shared embedding space by interlacing the embeddings of an image and its associated caption text [89, 122]. At least as interesting as using such sequence/structure embeddings as input for prediction will be the generative capability of such models that was investigated only recently in the form of ProtGPT2 [123]. These undirected generative approaches offer a perfect interplay with the fast, yet reliable predictions provided by predictors building on top of the proposed embeddings. Taken together, the two approaches might allow to generate and rank candidate proteins for a specific task completely in-silico which saves time and money when leveraging nature's most versatile work horses: proteins.

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., et al. Attention Is All You Need. In *Advances in neural information processing systems*, pages 5998–6008. 2017.

[2] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[4] Marx, V. Method of the year: protein structure prediction. *Nature methods*, 19(1):5–10, 2022.

[5] Rost, B. and Sander, C. Bridging the protein sequence-structure gap by structure predictions. *Annual review of biophysics and biomolecular structure*, 25(1):113–136, 1996.

[6] Steinegger, M. and Söding, J. Clustering huge protein sequence sets in linear time. *Nature Communications*, 9(1):1–8, 2018.

[7] Steinegger, M., Mirdita, M., and Söding, J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nature Methods*, 16(7):603–606, 2019.

[8] Crick, F. H. On protein synthesis. In *Symp Soc Exp Biol*, volume 12, page 8. 1958.

[9] Anfinsen, C. B. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

[10] Laskey, R., Honda, B., Mills, A., and Finch, J. Nucleosomes are assembled by an acidic protein which binds histones and transfers them to dna. *Nature*, 275(5679):416–420, 1978.

[11] Karplus, M. and Weaver, D. L. Protein-folding dynamics. *Nature*, 260(5550):404–406, 1976.

[12] Kabsch, W. and Sander, C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers: Original Research on Biomolecules*, 22(12):2577–2637, 1983.

[13] Rost, B. and Sander, C. Prediction of protein secondary structure at better than 70% accuracy. *Journal of molecular biology*, 232(2):584–599, 1993.

[14] Cuff, J. A., Clamp, M. E., Siddiqui, A. S., Finlay, M., and Barton, G. J. Jpred: a consensus secondary structure prediction server. *Bioinformatics (Oxford, England)*, 14(10):892–893, 1998.

[15] Jones, D. T. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of molecular biology*, 292(2):195–202, 1999.

[16] Klausen, M. S., Jespersen, M. C., Nielsen, H., Jensen, K. K., Jurtz, V. I., et al. Netsurfp-2.0: Improved prediction of protein structural features by integrated deep learning. *Proteins: Structure, Function, and Bioinformatics*, 87(6):520–527, 2019.

[17] Orengo, C. A., Flores, T., Taylor, W., and Thornton, J. M. Identification and classification of protein fold families. *Protein Engineering, Design and Selection*, 6(5):485–500, 1993.

[18] Sillitoe, I., Bordin, N., Dawson, N., Waman, V. P., Ashford, P., et al. Cath: increased structural coverage of functional space. *Nucleic acids research*, 49(D1):D266–D273, 2021.

[19] Orengo, C. A., Jones, D. T., and Thornton, J. M. Protein superfamilles and domain superfolds. *Nature*, 372(6507):631–634, 1994.

[20] Das, S., Lee, D., Sillitoe, I., Dawson, N. L., Lees, J. G., et al. Functional classification of cath superfamilies: a domain-based approach for protein function annotation. *Bioinformatics*, 31(21):3460–3467, 2015.

[21] Mirdita, M., Schütze, K., Moriwaki, Y., Heo, L., Ovchinnikov, S., et al. Colabfold - making protein folding accessible to all. *bioRxiv*, 2022. 10.1101/2021.08.15.456425.

[22] Nakai, K. and Kanehisa, M. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14(4):897–911, 1992.

[23] Chou, K.-C. and Elrod, D. W. Protein subcellular location prediction. *Protein engineering*, 12(2):107–118, 1999.

[24] Briesemeister, S., Rahnenfï¿ 1/2hrer, J. r., and Kohlbacher, O. Yloc—an interpretable web server for predicting subcellular localization. *Nucleic acids research*, 38(suppl_2):W497–W502, 2010.

[25] Goldberg, T., Hamp, T., and Rost, B. Loctree2 predicts localization for all domains of life. *Bioinformatics*, 28(18):i458–i465, 2012.

[26] Stärk, H., Dallago, C., Heinzinger, M., and Rost, B. Light attention predicts protein location from the language of life. *Bioinformatics Advances*, 1(1):vbab035, 2021.

[27] Thumuluri, V., Almagro Armenteros, J. J., Johansen, A. R., Nielsen, H., and Winther, O. Deeploc 2.0: multi-label subcellular localization prediction using protein language models. *Nucleic Acids Research*, 2022.

[28] Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., et al. Prottrans: Towards cracking the language of lifes code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[29] Jeffery, C. J. Moonlighting proteins. *Trends in biochemical sciences*, 24(1):8–11, 1999.

[30] Almagro Armenteros, J. J., Sønderby, C. K., Sønderby, S. K., Nielsen, H., and Winther, O. DeepLoc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, 33(21):3387–3395, 2017.

[31] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., et al. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[32] Jensen, L. J., Gupta, R., Staerfeldt, H.-H., and Brunak, S. Prediction of human protein function according to gene ontology categories. *Bioinformatics*, 19(5):635–642, 2003.

[33] Hawkins, T., Chitale, M., Luban, S., and Kihara, D. Pfp: Automated prediction of gene ontology functional annotations with confidence scores using protein sequence data. *Proteins: Structure, Function, and Bioinformatics*, 74(3):566–582, 2009.

[34] Done, B., Khatri, P., Done, A., and Draghici, S. Predicting novel human gene ontology annotations using semantic analysis. *IEEE/ACM transactions on computational biology and bioinformatics*, 7(1):91–99, 2008.

[35] Kulmanov, M., Khan, M. A., and Hoehndorf, R. Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, 2018.

[36] Zhao, Y., Fu, G., Wang, J., Guo, M., and Yu, G. Gene function prediction based on gene ontology hierarchy preserving hashing. *Genomics*, 111(3):334–342, 2019.

[37] Littmann, M., Heinzinger, M., Dallago, C., Olenyi, T., and Rost, B. Embeddings from deep learning transfer GO annotations beyond homology. *Scientific Reports*, 11(1):2045–2322, 2021. doi:10.1038/s41598-020-80786-0.

[38] Zhou, N., Jiang, Y., Bergquist, T. R., Lee, A. J., Kacsoh, B. Z., et al. The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome biology*, 20(1):1–23, 2019.

[39] Rost, B. Protein structures sustain evolutionary drift. *Folding and Design*, 2:S19–S24, 1997.

[40] Ofran, Y. and Rost, B. Protein–protein interaction hotspots carved into sequences. *PLoS computational biology*, 3(7):e119, 2007.

[41] Ma, B., Elkayam, T., Wolfson, H., and Nussinov, R. Protein–protein interactions: structurally conserved residues distinguish between binding sites and exposed protein surfaces. *Proceedings of the National Academy of Sciences*, 100(10):5772–5777, 2003.

[42] Berezin, C., Glaser, F., Rosenberg, J., Paz, I., Pupko, T., et al. Conseq: the identification of functionally and structurally important residues in protein sequences. *Bioinformatics*, 20(8):1322–1324, 2004.

[43] Glaser, F., Pupko, T., Paz, I., Bell, R. E., Bechor-Shental, D., et al. Consurf: identification of functional regions in proteins by surface-mapping of phylogenetic information. *Bioinformatics*, 19(1):163–164, 2003.

[44] Hudis, C. A. Trastuzumab—mechanism of action and use in clinical practice. *New England journal of medicine*, 357(1):39–51, 2007.

[45] Fowler, D. M. and Fields, S. Deep mutational scanning: a new style of protein science. *Nature methods*, 11(8):801–807, 2014.

[46] Simm, S., Einloft, J., Mirus, O., and Schleiff, E. 50 years of amino acid hydrophobicity scales: revisiting the capacity for peptide classification. *Biological research*, 49(1):1–19, 2016.

[47] Peters, C. and Elofsson, A. Why is the biological hydrophobicity scale more accurate than earlier experimental hydrophobicity scales? *Proteins: Structure, Function, and Bioinformatics*, 82(9):2190–2198, 2014.

[48] Sobel, I. and Feldman, G. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Project in*, pages 271–272, 1968.

[49] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., et al. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[50] Henikoff, S. and Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, 1992.

[51] Steinegger, M. and Söding, J. Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017.

[52] Baum, L. E. and Petrie, T. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[53] Remmert, M., Biegert, A., Hauser, A., and Söding, J. Hhblits: lightning-fast iterative protein sequence searching by hmm-hmm alignment. *Nature methods*, 9(2):173–175, 2012.

[54] Tien, M. Z., Meyer, A. G., Sydykova, D. K., Spielman, S. J., and Wilke, C. O. Maximum allowed solvent accessibilites of residues in proteins. *PloS one*, 8(11):e80635, 2013.

[55] Sander, C. and Schneider, R. Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins: Structure, Function, and Bioinformatics*, 9(1):56–68, 1991.

[56] Rost, B. Twilight zone of protein sequence alignments. *Protein engineering*, 12(2):85–94, 1999.

[57] Gribskov, M., McLachlan, A. D., and Eisenberg, D. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences*, 84(13):4355–4358, 1987.

[58] Marks, D. S., Colwell, L. J., Sheridan, R., Hopf, T. A., Pagnani, A., et al. Protein 3D structure computed from evolutionary sequence variation. *PLOS ONE*, 6(12):e28766, 2011.

[59] Wu, F.-Y. The potts model. *Reviews of modern physics*, 54(1):235, 1982.

[60] Marks, D. S., Colwell, L. J., Sheridan, R., Hopf, T. A., Pagnani, A., et al. Protein 3d structure computed from evolutionary sequence variation. *PloS one*, 6(12):e28766, 2011.

[61] Golkov, V., Skwark, M. J., Golkov, A., Dosovitskiy, A., Brox, T., et al. Protein contact prediction from amino acid co-evolution using convolutional networks for graph-valued images. *Advances in Neural Information Processing Systems*, 29, 2016.

[62] Wang, S., Li, W., Liu, S., and Xu, J. Raptorx-property: a web server for protein structure property prediction. *Nucleic acids research*, 44(W1):W430–W435, 2016.

[63] Hopf, T. A., Schärfe, C. P., Rodrigues, J. P., Green, A. G., Kohlbacher, O., et al. Sequence co-evolution gives 3d contacts and structures of protein complexes. *elife*, 3:e03430, 2014.

[64] Hopf, T. A., Ingraham, J. B., Poelwijk, F. J., Schärfe, C. P., Springer, M., et al. Mutation effects predicted from sequence co-variation. *Nature biotechnology*, 35(2):128–135, 2017.

[65] Schelling, M., Hopf, T. A., and Rost, B. Evolutionary couplings and sequence variation effect predict protein binding sites. *Proteins: Structure, Function, and Bioinformatics*, 86(10):1064–1074, 2018.

[66] Jones, D. T., Buchan, D. W., Cozzetto, D., and Pontil, M. Psicov: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, 28(2):184–190, 2012.

[67] Buel, G. R. and Walters, K. J. Can alphafold2 predict the impact of missense mutations on structure? *Nature Structural & Molecular Biology*, 29(1):1–2, 2022.

[68] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[69] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[70] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 2016.

[71] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[72] Asgari, E. and Mofrad, M. R. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PLOS ONE*, 10(11):e0141287, 2015.

[73] Boutet, E., Lieberherr, D., Tognolli, M., and Bairoch, A. UniProtKB/Swiss-Prot. *Methods in Molecular Biology*, 406:89–112, 2007.

[74] Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., et al. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

[75] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., et al. Language models are few-shot learners. 2020. 10.48550/ARXIV.2005.14165.

[76] Borgeaud, S., Mensch, A., Hoffmann, J., Cai, T., Rutherford, E., et al. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*, 2021.

[77] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543. 2014.

[78] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, 5:135–146, 2017.

[79] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., et al. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[80] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[81] Heinzinger, M., Elnaggar, A., Wang, Y., Dallago, C., Nechaev, D., et al. Modeling aspects of the language of life through transfer-learning protein sequences. *BMC bioinformatics*, 20(1):1–17, 2019.

[82] Suzek, B. E., Huang, H., McGarvey, P., Mazumder, R., and Wu, C. H. Uniref: comprehensive and non-redundant uniprot reference clusters. *Bioinformatics*, 23(10):1282–1288, 2007.

[83] The UniProt Consortium. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Research*, 47(D1):D506–D515, 2019.

[84] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[85] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[86] Fuchs, F., Worrall, D., Fischer, V., and Welling, M. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

[87] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[88] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. Improving language understanding by generative pre-training. 2018.

[89] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[90] Taylor, W. L. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

[91] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., et al. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[92] Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[93] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[94] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., et al. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[95] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., et al. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[96] Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021.

[97] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using aßiamesettime delay neural network. *Advances in neural information processing systems*, 6, 1993.

[98] Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

[99] Hermans, A., Beyer, L., and Leibe, B. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[100] van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[101] Lu, A. X., Zhang, H., Ghassemi, M., and Moses, A. Self-supervised contrastive learning of protein representations by mutual information maximization. *BioRxiv*, 2020.

[102] Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., et al. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020.

[103] Marquet, C., Heinzinger, M., Olenyi, T., Dallago, C., Erckert, K., et al. Embeddings from protein language models predict conservation and variant effects. *Human genetics*, pages 1–19, 2021.

[104] Heinzinger, M., Littmann, M., Sillitoe, I., Bordin, N., Orengo, C., et al. Contrastive learning on protein embeddings enlightens midnight zone. *NAR Genomics and Bioinformatics*, 4(2), 2022. ISSN 2631-9268. 10.1093/nargab/lqac043. Lqac043.

[105] Rao, R. M., Liu, J., Verkuil, R., Meier, J., Canny, J., et al. Msa transformer. In *International Conference on Machine Learning*, pages 8844–8856. PMLR, 2021.

[106] Perdigão, N., Heinrich, J., Stolte, C., Sabir, K. S., Buckley, M. J., et al. Unexpected features of the dark proteome. *Proceedings of the National Academy of Sciences*, 112(52):15898–15903, 2015.

[107] Dallago, C., Schütze, K., Heinzinger, M., Olenyi, T., Littmann, M., et al. Learned embeddings from deep learning to visualize and predict protein sets. *Current Protocols*, 1(5):e113, 2021.

[108] Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Research*, 47(D1):D330–D338, 2019.

[109] El-Mabrouk, N. and Slonim, D. K. ISMB 2020 proceedings. *Bioinformatics*, 36(Supplement_1):i1–i2, 2020. 10.1093/bioinformatics/btaa537.

[110] Bernhofer, M., Dallago, C., Karl, T., Satagopam, V., Heinzinger, M., et al. Predictprotein-predicting protein structure and function for 29 years. *Nucleic acids research*, 49(W1):W535–W540, 2021.

[111] Reeb, J., Wirth, T., and Rost, B. Variant effect predictions capture some aspects of deep mutational scanning experiments. *BMC bioinformatics*, 21(1):1–12, 2020.

[112] Meier, J., Rao, R., Verkuil, R., Liu, J., Sercu, T., et al. Language models enable zero-shot prediction of the effects of mutations on protein function. *Advances in Neural Information Processing Systems*, 34, 2021.

[113] Riesselman, A. J., Ingraham, J. B., and Marks, D. S. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018.

[114] Laine, E., Karami, Y., and Carbone, A. Gemme: a simple and fast global epistatic model predicting mutational effects. *Molecular biology and evolution*, 36(11):2604–2619, 2019.

[115] Ballas, S. K., Gupta, K., and Adams-Graves, P. Sickle cell pain: a critical reappraisal. *Blood, The Journal of the American Society of Hematology*, 120(18):3647–3656, 2012.

[116] Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[117] Littmann, M., Bordin, N., Heinzinger, M., Schütze, K., Dallago, C., et al. Clustering funfams using sequence embeddings improves ec purity. *Bioinformatics*, 37(20):3449–3455, 2021.

[118] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[119] Tay, Y., Tran, V. Q., Dehghani, M., Ni, J., Bahri, D., et al. Transformer memory as a differentiable search index. 2022. 10.48550/ARXIV.2202.06991.

[120] Varadi, M., Anyango, S., Deshpande, M., Nair, S., Natassia, C., et al. AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic Acids Research*, 50(D1):D439–D444, 2021. ISSN 0305-1048. 10.1093/nar/gkab1061.

[121] Hsu, C., Verkuil, R., Liu, J., Lin, Z., Hie, B., et al. Learning inverse folding from millions of predicted structures. *bioRxiv*, 2022. 10.1101/2022.04.10.487779.

[122] Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., et al. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*, 2022.

[123] Ferruz, N., Schmidt, S., and Höcker, B. A deep unsupervised language model for protein design. *bioRxiv*, 2022.

A. Appendix

List of Publications

The publication-based dissertation at hand is based on the following four peer-reviewed and published manuscripts:

- **Michael Heinzinger**, Ahmed Elnaggar, Yu Wang, Christian Dallago, Dmitrii Nechaev, Florian Matthes, and Burkhard Rost. *Modeling aspects of the language of life through transfer-learning protein sequences.* BMC bioinformatics 20, no. 1 (2019): 1-17.
- Maria Littmann, **Michael Heinzinger**, Christian Dallago, Tobias Olenyi, and Burkhard Rost. *Embeddings from deep learning transfer GO annotations beyond homology.* Scientific reports 11, no. 1 (2021): 1-14.
- Céline Marquet, **Michael Heinzinger**, Tobias Olenyi, Christian Dallago, Kyra Erckert, Michael Bernhofer, Dmitrii Nechaev, and Burkhard Rost. *Embeddings from protein language models predict conservation and variant effects.* Human genetics (2021): 1-19.
- **Michael Heinzinger**, Maria Littmann, Ian Sillitoe, Nicola Bordin, Christine Orengo, and Burkhard Rost. *Contrastive learning on protein embeddings enlightens midnight zone at lightning speed.* NAR Genomics and Bioinformatics 4.2 (2022).

In addition to the publications above, I co-main-authored the following peer-reviewed publication that are not discussed in this dissertation:

- Ahmed Elnaggar and **Michael Heinzinger**, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik and Burkhard Rost. *ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing,* in IEEE Transactions on Pattern Analysis and Machine Intelligence, doi: 10.1109/TPAMI.2021.3095381.
- Dagmar Ilzhöfer and **Michael Heinzinger** and Burkhard Rost. *SETH predicts nuances of residue disorder from protein embeddings,* in Front Bioinform. 2022;2:1019597, doi:10.3389/fbinf.2022.1019597.

Results discussed in Section 4 and 5 build on top of results presented in [28]. Also, I co-authored the following peer-reviewed publications that are not discussed in this dissertation but also build on top of the work introduced here:

- Christian Dallago, Konstantin Schütze, **Michael Heinzinger**, Tobias Olenyi, Maria Littmann, Amy X. Lu, Kevin K. Yang et al. *Learned embeddings from deep learning to visualize and predict protein sets.* Current Protocols 1, no. 5 (2021): e113.
- Hannes Stärk, Christian Dallago, **Michael Heinzinger**, and Burkhard Rost. *Light attention predicts protein location from the language of life.* Bioinformatics Advances 1, no. 1 (2021): vbab035.
- Maria Littmann, Nicola Bordin, **Michael Heinzinger**, Konstantin Schütze, Christian Dallago, Christine Orengo, and Burkhard Rost. *Clustering FunFams using sequence embeddings improves EC purity.* Bioinformatics 37, no. 20 (2021): 3449-3455.
- Maria Littmann, **Michael Heinzinger**, Christian Dallago, Konstantin Weissenow, and Burkhard Rost. *Protein embeddings and deep learning predict binding residues for various ligand classes.* Scientific reports 11, no. 1 (2021): 1-15.
- Konstantin Weissenow, **Heinzinger, Michael**, Rost, Burkhard. *Protein language-model embeddings for fast, accurate, and alignment-free protein structure prediction.* Structure (2022): 0969-2126 doi: 10.1016/j.str.2022.05.001

Additionally, I have co-authored the following peer-reviewed publications that are not discussed in this dissertation:

- Andrea Schafferhans, Seán I. O'Donoghue, **Michael Heinzinger**, and Burkhard Rost. *Dark proteins important for cellular function.* Proteomics 18, no. 21-22 (2018): 1800227.
- Jiajun Qiu, Michael Bernhofer, **Michael Heinzinger**, Sofie Kemper, Tomas Norambuena, Francisco Melo, and Burkhard Rost. *ProNA2020 predicts protein–DNA, protein–RNA, and protein–protein binding proteins and residues from sequence.* Journal of molecular biology 432, no. 7 (2020): 2428-2443.
- Jan Zaucha, **Michael Heinzinger**, Svetlana Tarnovskaya, Burkhard Rost, and Dmitrij Frishman. *Family-specific analysis of variant pathogenicity prediction tools.* NAR genomics and bioinformatics 2, no. 2 (2020): lqaa014.
- Jan Zaucha, **Michael Heinzinger**, A. Kulandaisamy, Evans Kataka, Óscar Llorian Salvádor, Petr Popov, Burkhard Rost, M. Michael Gromiha, Boris S. Zhorov, and Dmitrij Frishman. *Mutations in transmembrane proteins: diseases, evolutionary insights, prediction and comparison with globular proteins.* Briefings in Bioinformatics 22, no. 3 (2021): bbaa132.

- Michael Bernhofer, Christian Dallago, Tim Karl, Venkata Satagopam, **Michael Heinzinger**, Maria Littmann, Tobias Olenyi et al. *PredictProtein - predicting protein structure and function for 29 years.* Nucleic acids research 49, no. W1 (2021): W535-W540.
- Jana Weisser, Teresa Pohl, **Michael Heinzinger**, Natalia P. Ivleva, Thomas Hofmann, and Karl Glas. *The identification of microplastics based on vibrational spectroscopy data–a critical review of data analysis routines.* TrAC Trends in Analytical Chemistry (2022): 116535.

Also, I have co-authored the following publications that have been submitted to peer-review and have already been published as pre-print on bioRxiv. The results are not discussed in this dissertation.

- Ivan Koludarov, Mariana Velasque, Thomas Timm, Günter Lochnit, **Michael Heinzinger**, Andreas Vilcinskas, Rosalyn Gloag et al. *Bee core venom genes predominantly originated before aculeate stingers evolved.* bioRxiv (2022).
- Vamsi Nallapareddy, Nicola Bordin, Ian Sillitoe, **Michael Heinzinger**, Maria Littmann, Vaishali Waman, Neeladri Sen, Burkhard Rost, and Christine Orengo. *CATHe: Detection of remote homologues for CATH superfamilies using embeddings from protein language models.* bioRxiv (2022).
- Konstantin Schütze, **Heinzinger Michael**, Steinegger Martin and Rost Burkhard. *Nearest neighbor search on embeddings rapidly identifies distant protein relations.* bioRxiv (2022).
- Noelia Ferruz, **Michael Heinzinger**, Mehmet Akdel, Alexander Goncearenco, Luca Naef and Christian Dallago. *From sequence to function through structure: deep learning for protein design.* bioRxiv (2022)

A.1. Publications Included in This Dissertation

A.1.1. Heinzinger, Elnaggar *et al.*, BMC Bioinformatics (2019)

Heinzinger *et al. BMC Bioinformatics* (2019) 20:723
https://doi.org/10.1186/s12859-019-3220-8

BMC Bioinformatics

RESEARCH ARTICLE **Open Access**

Modeling aspects of the language of life through transfer-learning protein sequences

Michael Heinzinger[1,2*†], Ahmed Elnaggar[1,2†], Yu Wang[3], Christian Dallago[1,2], Dmitrii Nechaev[1,2], Florian Matthes[4] and Burkhard Rost[1,5,6,7]

Abstract

Background: Predicting protein function and structure from sequence is one important challenge for computational biology. For 26 years, most state-of-the-art approaches combined machine learning and evolutionary information. However, for some applications retrieving related proteins is becoming too time-consuming. Additionally, evolutionary information is less powerful for small families, e.g. for proteins from the *Dark Proteome*. Both these problems are addressed by the new methodology introduced here.

Results: We introduced a novel way to represent protein sequences as continuous vectors (*embeddings*) by using the language model ELMo taken from natural language processing. By modeling protein sequences, ELMo effectively captured the biophysical properties of the language of life from unlabeled big data (UniRef50). We refer to these new embeddings as *SeqVec* (*Seq*uence-to-*Vec*tor) and demonstrate their effectiveness by training simple neural networks for two different tasks. At the per-residue level, secondary structure (Q3 = 79% ± 1, Q8 = 68% ± 1) and regions with intrinsic disorder (MCC = 0.59 ± 0.03) were predicted significantly better than through one-hot encoding or through Word2vec-like approaches. At the per-protein level, subcellular localization was predicted in ten classes (Q10 = 68% ± 1) and membrane-bound were distinguished from water-soluble proteins (Q2 = 87% ± 1). Although *SeqVec* embeddings generated the best predictions from single sequences, no solution improved over the best existing method using evolutionary information. Nevertheless, our approach improved over some popular methods using evolutionary information and for some proteins even did beat the best. Thus, they prove to condense the underlying principles of protein sequences. Overall, the important novelty is speed: where the lightning-fast *HHblits* needed on average about two minutes to generate the evolutionary information for a target protein, *SeqVec* created embeddings on average in 0.03 s. As this speed-up is independent of the size of growing sequence databases, *SeqVec* provides a highly scalable approach for the analysis of big data in proteomics, i.e. microbiome or metaproteome analysis.

Conclusion: Transfer-learning succeeded to extract information from unlabeled sequence databases relevant for various protein prediction tasks. SeqVec modeled the language of life, namely the principles underlying protein sequences better than any features suggested by textbooks and prediction methods. The exception is evolutionary information, however, that information is not available on the level of a single sequence.

Keywords: Machine Learning, Language Modeling, Sequence Embedding, Secondary structure prediction, Localization prediction, Transfer Learning, Deep Learning

* Correspondence: mheinzinger@rostlab.org; assistant@rostlab.org
†Michael Heinzinger and Ahmed Elnaggar contributed equally to this work.
[1]Department of Informatics, Bioinformatics & Computational Biology - i12, TUM (Technical University of Munich), Boltzmannstr. 3, 85748 Garching/Munich, Germany
[2]TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany
Full list of author information is available at the end of the article

Background

The combination of evolutionary information (from Multiple Sequence Alignments – MSA) and Machine Learning/Artificial Intelligence (standard feed-forward artificial neural networks – ANN) completely changed protein secondary structure prediction [1–3]. The concept was quickly taken up [4–8] and predictions improved even more with larger families increasing evolutionary information through diversity [9, 10]. The idea was applied to other tasks, including the prediction of transmembrane regions [11–13], solvent accessibility [14], residue flexibility (B-values) [15, 16], inter-residue contacts [17] and protein disorder [15, 18–20]. Later, automatic methods predicting aspects of protein function improved by combining evolutionary information and machine learning, including predictions of subcellular localization (aka cellular compartment or CC in GO [21, 22]), protein interaction sites [23–25], and the effects of sequence variation upon function [26, 27]. Arguably, the most important breakthrough for protein structure prediction over the last decade was a more efficient way of using evolutionary couplings [28–31].

Although evolutionary information has increasingly improved prediction methods, it is also becoming increasingly costly. As sequencing becomes cheaper, the number of bio-sequence databases grow faster than computing power. For instance, the number of UniProt entries is now more than doubling every two years [32]. An all-against-all comparison executed to build up profiles of evolutionary information squares this number: every two years the job increases 4-fold while computer power grows less than 2-fold. Consequently, methods as fast as PSI-BLAST [33] have to be replaced by faster solutions such as HHblits [34]. Even its latest version HHblits3 [35] still needs several minutes to search UniRef50 (subset of UniProt) for a single query protein. The next step up in speed such as MMSeqs2 [36] appear to cope with the challenge at the expense of increasing hardware requirements while databases keep growing. However, even these solutions might eventually lose the battle against the speedup of sequencing. Analyzing data sets involving millions of proteins, i.e. samples of the human gut microbiota or metagenomic samples, have already become a major challenge [35]. Secondly, evolutionary information is still missing for some proteins, e.g. for proteins with substantial intrinsically disordered regions [15, 37, 38], or the entire *Dark Proteome* [39] full of proteins that are less-well studied but important for function [40].

Here, we propose a novel embedding of protein sequences that replaces the explicit search for evolutionary related proteins by an implicit transfer of biophysical information derived from large, unlabeled sequence data (here UniRef50). We adopted a method that has been revolutionizing Natural Language Processing (NLP), namely the bi-directional language model ELMo (Embeddings from Language Models) [41]. In NLP, ELMo is trained on unlabeled text-corpora such as Wikipedia to predict the most probable next word in a sentence, given all previous words in this sentence. By learning a probability distribution for sentences, these models autonomously develop a notion for syntax and semantics of language. The trained vector representations (embeddings) are contextualized, i.e. the embeddings of a given word depend on its context. This has the advantage that two identical words can have different embeddings, depending on the words surrounding them. In contrast to previous non-contextualized approaches such as word2-vec [42, 43], this allows to take the ambiguous meaning of words into account.

We hypothesized that the ELMo concept could be applied to model protein sequences. Three main challenges arose. (1) Proteins range from about 30 to 33,000 residues, a much larger range than for the average English sentence extending over 15–30 words [44], and even more extreme than notable literary exceptions such as James Joyce's Ulysses (1922) with almost 4000 words in a sentence. Longer proteins require more GPU memory and the underlying models (so-called LSTMs: Long Short-Term Memory networks [45]) have only a limited capability to remember long-range dependencies. (2) Proteins mostly use 20 standard amino acids, 100,000 times less tokens than in the English language. Smaller vocabularies might be problematic if protein sequences encode a similar complexity as sentences. (3) We found UniRef50 to contain almost ten times more tokens (9.5 billion amino acids) than the largest existing NLP corpus (1 billion words). Simply put: Wikipedia is roughly ten times larger than Webster's Third New International Dictionary and the entire UniProt is over ten times larger than Wikipedia. As a result, larger models might be required to absorb the information in biological databases.

We trained ELMo on UniRef50 and assessed the predictive power of the embeddings by application to tasks on two levels: per-residue (word-level) and per-protein (sentence-level). For the per-residue prediction task, we predicted secondary structure and long intrinsic disorder. For the per-protein prediction task, we predicted subcellular localization and trained a classifier distinguishing between membrane-bound and water-soluble proteins. We used publicly available data sets from two recent methods that achieved break-through performance through Deep Learning, namely NetSurfP-2.0 for secondary structure [46] and DeepLoc for localization [47]. We compared the performance of the *SeqVec* embeddings to state-of-the-art methods using evolutionary information, and also to a popular embedding tool for protein sequences originating from the Word2vec approach, namely *ProtVec* [42]. Notably, while *ProtVec* captures local information, it loses information on sequence ordering, and the resulting residue embeddings are insensitive to their context (non-contextualized), i.e. the same word results in the same embedding regardless of the specific context.

Understanding a language typically implies to understand most typical constructs convened in that language. Modeling a language in a computer can have many meanings, spanning from the automatic understanding of the semantic of languages, to parsing some underlying rules of a language (e.g. syntax). Arguably, proteins are the most important machinery of life. Protein sequence largely determines protein structure, which somehow determines protein function [48]. Thus, the expression of the language of life are essentially protein sequences. Understanding those sequences implies to predict protein structure from sequence. Despite recent successes [49, 50], this is still not possible for all proteins. However, the novel approach introduced here succeeds to model protein sequences in the sense that it implicitly extracts grammar-like principles (as embeddings) which are much more successful in predicting aspects of protein structure and function than any of the biophysical features previously used to condensate expert knowledge of protein folding, or any other previously tried simple encoding of protein sequences.

Results

Modeling protein sequences through SeqVec embeddings

SeqVec, our ELMo-based implementation, was trained for three weeks on 5 Nvidia Titan GPUs with 12 GB memory each. The model was trained until its *perplexity* (uncertainty when predicting the next token) converged at around 10.5 (Additional file 1: Figure S1). Training and testing were not split due to technical limitations (incl. CPU/GPU). ELMo was designed to reduce the risk of overfitting by sharing weights between forward and backward LSTMs and by using dropout. The model had about 93 M (mega/million) free parameters compared to the 9.6G (giga/billion) tokens to predict leading to a ratio of samples/free parameter below 1/100, the best our group has ever experienced in a prediction task. Similar approaches have shown that even todays largest models (750 M free parameters) are not able to overfit on a large corpus (250 M protein sequences) [51].

SeqVec embeddings appeared robust

When training ELMo on SWISS-PROT (0.5 M sequences), we obtained less useful models, i.e. the subsequent prediction methods based on those embeddings were less accurate. Training on UniRef50 (33 M sequences) gave significantly better results in subsequent supervised prediction tasks, and we observed similar results when using different hyperparameters. For instance, increasing the number of LSTM layers in ELMo (from two to four) gave a small, non-significant improvement. As the expansion of 2 to 4 layers roughly doubled time for training and retrieving embeddings, we decided to trade speed for insignificant improvement and continued with the faster two-layer ELMo architecture. Computational limitations hindered us from fully completeing the modelling of UniRef90 (100 million sequences). Nevertheless, after four weeks of training, the models neither appeared to be better nor significantly worse than those for UniRef50. Users of the embeddings need to be aware that every time a new ELMo model is trained, the downstream supervised prediction method needs to be retrained in the following sense. Assume we transfer-learn UniRef50 through SeqVec1, then use SeqVec1 to machine learn DeepSeqVec1 for a supervised task (e.g. localization prediction). In a later iteration, we redo the transfer learning with different hyperparameters to obtain SeqVec2. For any given sequence, the embeddings of SeqVec2 will differ from those of SeqVec1, as a result, passing embeddings derived from SeqVec2 to DeepSeqVec1 will not provide meaningful predictions.

Per-residue performance high, not highest

NetSurfP-2.0 feeds HHblits or MMseqs2 profiles into advanced combinations of Deep Learning architectures [46] to predict secondary structure, reaching a three-state per-residue accuracy Q3 of 82–85% (lower value: small, partially non-redundant CASP12 set, upper value: larger, more redundant TS115 and CB513 sets; Table 1, Fig. 1; several contenders such as *Spider3* and *RaptorX* reach within three standard errors). All six methods developed by us fell short of reaching this mark, both methods not using evolutionary information/profiles (DeepSeqVec, DeepProtVec, DeepOneHot, DeepBLOSUM65), but also those that did use profiles (*DeepProf*, DeepProf+SeqVec, Fig. 1a, Table 1). The logic in our acronyms was as follows (Methods): "*Prof*" implied using profiles (evolutionary information), *SeqVec* (Sequence-to-Vector) described using pre-trained ELMo embeddings, "Deep" before the method name suggested applying a simple deep learning method trained on particular prediction tasks using SeqVec embeddings only (DeepSeqVec), profiles without (DeepProf) or with embeddings (DeepProf+SeqVec), or other simple encoding schema (ProtVec, OneHot or sparse encoding, or BLOSUM65). When comparing methods that use only single protein sequences as input (DeepSeqVec, DeepProtVec, DeepOneHot, DeepBLOSUM65; all white in Table 1), the new method introduced here, *SeqVec* outperformed others not using profiles by three standard errors (*P*-value< 0.01; Q3: 5–10 percentage points, Q8: 5–13 percentage points, MCC:

Table 1 Per-residue predictions: secondary structure and disorder

Data	*Prediction task*	*Secondary structure*		*Disorder*	
	Method	*Q3 (%)*	*Q8 (%)*	*MCC*	*FPR*
CASP12	*NetSurfP-2.0 (hhblits)*[a,b]	**82.4**	**71.1**	0.604	**0.011**
	NetSurfP-1.0[a,b]	70.9	–	–	–
	Spider3[a,b]	79.1	–	0.582	0.026
	RaptorX[a,b]	78.6	66.1	**0.621**	0.045
	Jpred4[a,b]	76.0	–	–	–
	DeepSeqVec	73.1 ± 1.3	61.2 ± 1.6	0.575 ± 0.075	0.026 ± 0.008
	DeepProf[b]	76.4 ± 2.0	62.7 ± 2.2	0.506 ± 0.057	0.022 ± 0.009
	DeepProf + SeqVec[b]	76.5 ± 1.5	64.1 ± 1.5	0.556 ± 0.080	0.022 ± 0.008
	DeepProtVec	62.8 ± 1.7	50.5 ± 2.4	0.505 ± 0.064	0.016 ± 0.006
	DeepOneHot	67.1 ± 1.6	54.2 ± 2.1	0.461 ± 0.064	0.012 ± 0.005
	DeepBLOSUM65	67.0 ± 1.6	54.5 ± 2.0	0.465 ± 0.065	0.012 ± 0.005
TS115	*NetSurfP-2.0 (hhblits)*[a,b]	**85.3**	**74.4**	**0.663**	**0.006**
	NetSurfP-1.0[a,b]	77.9	–	–	–
	Spider3[a,b]	83.9	–	0.575	0.008
	RaptorX[a,b]	82.2	71.6	0.567	0.027
	Jpred4[a,b]	76.7	–	–	–
	DeepSeqVec	79.1 ± 0.8	67.6 ± 1.0	0.591 ± 0.028	0.012 ± 0.001
	DeepProf[b]	81.1 ± 0.6	68.3 ± 0.9	0.516 ± 0.028	0.012 ± 0.002
	DeepProf + SeqVec[b]	82.4 ± 0.7	70.3 ± 1.0	0.585 ± 0.029	0.013 ± 0.003
	DeepProtVec	66.0 ± 1.0	54.4 ± 1.3	0.470 ± 0.028	0.011 ± 0.002
	DeepOneHot	70.1 ± 0.8	58.5 ± 1.1	0.476 ± 0.028	0.008 ± 0.001
	Deep BLOSUM65	70.3 ± 0.8	58.1 ± 1.1	0.488 ± 0.029	0.007 ± 0.001
CB513	*NetSurfP-2.0 (hhblits)*[a,b]	**85.3**	**72.0**	–	–
	NetSurfP-1.0[a,b]	78.8	–	–	–
	Spider3[a,b]	84.5	–	–	–
	RaptorX[a,b]	82.7	70.6	–	–
	Jpred4[a,b]	77.9	–	–	–
	DeepSeqVec	76.9 ± 0.5	62.5 ± 0.6	–	–
	DeepProf[b]	80.2 ± 0.4	64.9 ± 0.5	–	–
	DeepProf + SeqVec[b]	80.7 ± 0.5	66.0 ± 0.5	–	–
	DeepProtVec	63.5 ± 0.4	48.9 ± 0.5	–	–
	DeepOneHot	67.5 ± 0.4	52.9 ± 0.5	–	–
	DeepBLOSUM65	67.4 ± 0.4	53.0 ± 0.5	–	–

Performance comparison for secondary structure (3- vs. 8-classes) and disorder prediction (binary) for the CASP12, TS115 and CB513 data sets. Accuracy (Q3, Q10) is given in percentage. Results marked by [a] are taken from NetSurfP-2.0 [46]; the authors did not provide standard errors. Highest numerical values in each column in bold letters. Methods DeepSeqVec, DeepProtVec, DeepOneHot and DeepBLOSUM65 use only information from single protein sequences. Methods using evolutionary information (MSA profiles) are marked by [b]; these performed best throughout

0.07–0.12, Table 1). Using a context-independent language model derived from the Word2vec approach, namely DeepProtVec was worse by 10 percentage points (almost six standard errors). On the other hand, our implementation of evolutionary information (DeepProf using HHblits profiles) remained about 4–6 percentage points below NetSurfP-2.0 (Q3 = 76–81%, Fig. 1, Table 1). Depending on the test set, using *SeqVec* embeddings instead of evolutionary information (DeepSeqVec: Fig. 1a, Table 1) remained 2–3 percentage points below that mark (Q3 = 73–79%, Fig. 1a, Table 1). Using both evolutionary information and *SeqVec* embeddings (DeepProf+SeqVec) improved over both, but still did not reach the top (Q3 = 77–82%). In fact, the ELMo embeddings alone (DeepSeqVec) did not surpass any of the best methods using evolutionary information tested on the same data set (Fig. 1a).

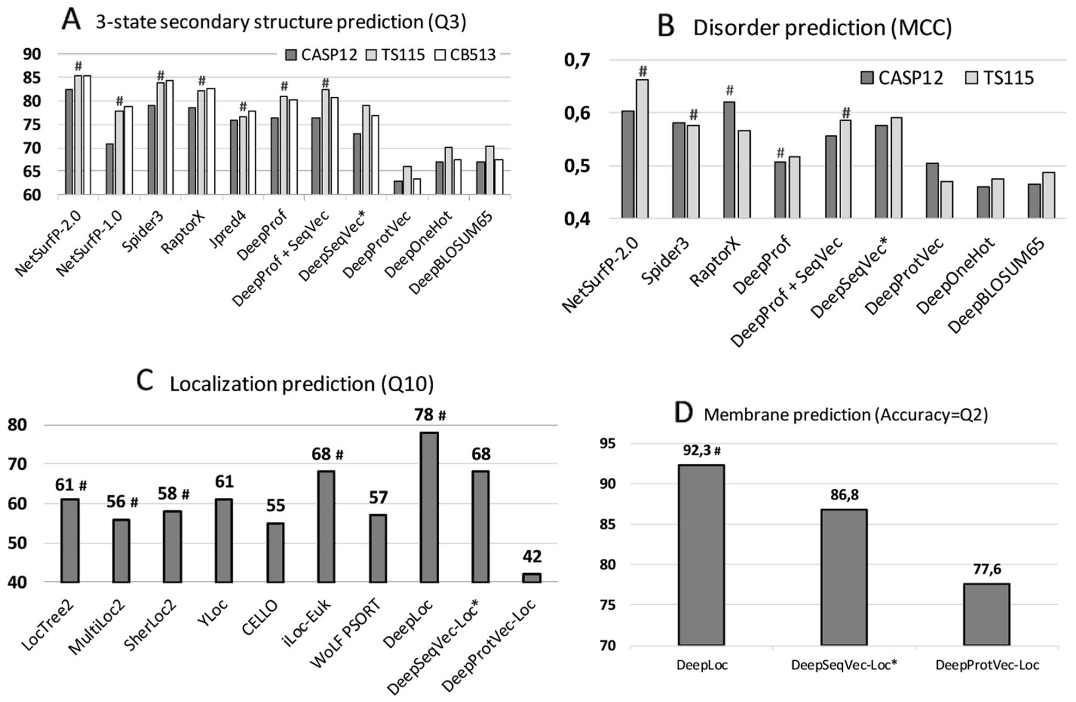

Fig. 1 Performance comparisons. The predictive power of the ELMo-based SeqVec embeddings was assessed for per-residue (upper row) and per-protein (lower row) prediction tasks. Methods using evolutionary information are highlighted by hashes above the bars. Approaches using only the proposed *SeqVec* embeddings are highlighted by stars after the method name. Panel **A** used three different data sets (CASP12, TS115, CB513) to compare three-state secondary structure prediction (y-axis: Q3; all DeepX developed here to test simple deep networks on top of the encodings tested; DeepProf used evolutionary information). Panel **B** compared predictions of intrinsically disordered residues on two data sets (CASP12, TS115; y-axis: MCC). Panel **C** compared per-protein predictions for subcellular localization between top methods (numbers for Q10 taken from DeepLoc [47]) and embeddings based on single sequences (Word2vec-like *ProtVec* [42] and our ELMo-based *SeqVec*). Panel **D**: the same data set was used to assess the predictive power of SeqVec for the classification of a protein into membrane-bound and water-soluble

For the prediction of intrinsic disorder, we observed the same: NetSurfP-2.0 performed best; our implementation of evolutionary information (DeepProf) performed worse (Fig. 1b, Table 1). However, for this task the embeddings alone (DeepSeqVec) performed relatively well, exceeding our in-house implementation of a model using evolutionary information (DeepSeqVec MCC = 0.575–0.591 vs. DeepProf MCC = 0.506–0.516, Table 1). The combination of evolutionary information and embeddings (DeepProf+SeqVec) improved over using evolutionary information alone but did not improve over the *SeqVec* embeddings for disorder. Compared to other methods, the embeddings alone reached similar values (Fig. 1b).

Per-protein performance close to best

For predicting subcellular localization (cellular compartments) in ten classes, *DeepLoc* [47] is top with Q10 = 78% (Fig. 1c, Table 2). For simplicity, we only tested methods not using evolutionary information/profiles for this task. Our sequence-only embeddings model DeepSeqVec-Loc reached second best performance together with iLoc-Euk [52] at Q10 = 68% (Fig. 1c, Table 2). Unlike the per-residue predictions, for this application the SeqVec embeddings outperformed several popular prediction methods that use evolutionary information by up to 13 percentage points in Q10 (Table 2: DeepSeqVec-Loc vs. methods shown in grayed rows). The gain of the context-dependent SeqVec model introduced here over context-independent versions such as ProtVec (from Word2vec) was even more pronounced than for the per-residue prediction task (Q10 68 ± 1% vs. 42 ± 1%).

Performance for the classification into membrane-bound and water-soluble proteins followed a similar trend (Fig. 1d, Table 2): while DeepLoc still performed best (Q2 = 92.3, MCC = 0.844), DeepSeqVec-Loc reached just a few percentage points lower (Q2 = 86.8 ± 1.0,

Table 2 Per-protein predictions: localization and membrane/globular

Method	*Localization*		*Membrane/globular*	
	Q10 (%)	*Gorodkin (MCC)*	*Q2*	*MCC*
LocTree2[a,b]	61	0.53		
MultiLoc2[a,b]	56	0.49		
CELLO[a]	55	0.45		
WoLF PSORT[a]	57	0.48		
YLoc[a]	61	0.53		
SherLoc2[a,b]	58	0.51		
iLoc-Euk[a,b]	68	0.64		
DeepLoc[a,b]	**78**	**0.73**	**92.3**	**0.844**
DeepSeqVec-Loc	68 ± 1	0.61 ± 0.01	86.8 ± 1.0	0.725 ± 0.021
DeepProtVec-Loc	42 ± 1	0.19 ± 0.01	77.6 ± 1.3	0.531 ± 0.026

Performance for per-protein prediction of subcellular localization and classifying proteins into membrane-bound and water-soluble. Results marked by [a] taken from DeepLoc [47]; the authors provided no standard errors. The results reported for *SeqVec* and *ProtVec* were based on single protein sequences, i.e. methods NOT using evolutionary information (neither during training nor testing). All methods using evolutionary information are marked by [b]; best in each set marked by bold numbers

MCC = 0.725 ± 0.021; full confusion matrix Additional file 1: Figure S2). In contrast to this, ProtVec, another method using only single sequences, performed substantially worse (Q2 = 77.6 ± 1.3, MCC = 0.531 ± 0.026).

Visualizing results

Lack of insight often triggers the misunderstanding that machine learning methods are black box solutions barring understanding. In order to interpret the *SeqVec* embeddings, we have projected the protein-embeddings of the per-protein prediction data upon two dimensions using t-SNE [53]. We performed this analysis once for the raw embeddings (SeqVec, Fig. 2 upper row) and once for the hidden layer representation of the per-protein network (DeepSeqVec-Loc) after training (Fig. 2 lower row). All t-SNE representations in Fig. 2 were created using 3000 iterations and the cosine distance as metric. The two analyses differed only in that the perplexity was set to 20 for one (*SeqVec*) and 15 for the other (DeepSeqVec-Loc). The t-SNE representations were colored either according to their localization within the cell (left column of Fig. 2) or according to whether they are membrane-bound or water-soluble (right column).

Despite never provided during training, the raw embeddings appeared to capture some signal for classifying proteins by localization (Fig. 2, upper row, left column). The most consistent signal was visible for extra-cellular proteins. Proteins attached to the cell membrane or located in the endoplasmic reticulum also formed well-defined clusters. In contrast, the raw embeddings neither captured a consistent signal for nuclear nor for mitochondrial proteins. Through training, the network improved the signal to reliably classify mitochondrial and plastid proteins. However, proteins in the nucleus and cell membrane continued to be poorly distinguished via t-SNE.

Coloring the t-SNE representations for membrane-bound or water-soluble proteins (Fig. 2, right column), revealed that the raw embeddings already provided well-defined clusters although never trained on membrane prediction (Fig. 2, upper row). After training, the classification was even better (Fig. 2, lower row).

Analogously, we used t-SNE projections to analyze SeqVec embeddings on different levels of complexity inherent to proteins (Fig. 3), ranging from the building blocks (amino acids, Fig. 3a), to secondary structure defined protein classes (Fig. 3b), over functional features (Fig. 3c), and onto the macroscopic level of the kingdoms of life and viruses (Fig. 3d; classifications in panels 3b-3d based on SCOPe [54]). Similar to the results described in [51], our projection of the embedding space confirmed that the model successfully captured bio-chemical and bio-physical properties on the most fine-grained level, i.e. the 20 standard amino acids (Fig. 3a). For example, aromatic amino acids (W, F, Y) are well separated from aliphatic amino acids (A, I, L, M, V) and small amino acids (A, C, G, P, S, T) are well separated from large ones (F, H, R, W, Y). The projection of the letter indicating an unknown amino acid (X), clustered closest to the amino acids alanine (A) and glycine (G) (data not shown). Possible explanations for this could be that the two amino acids with the smallest side chains might be least biased towards other biochemical features like charge and that they are the 2nd (A) and 4th (G) most frequent amino acids in our training set (Additional file 1: Table S1). Rare (O, U) and ambiguous amino acids (Z, B) were removed from the projection as their clustering showed that the model could not learn reasonable embeddings from the very small number of samples.

High-level structural classes as defined in SCOPe (Fig. 3b) were also captured by SeqVec embeddings. Although the embeddings were only trained to predict the next amino acid in a protein sequence, well separated clusters emerged from those embeddings in structure space. Especially, membrane proteins and small proteins formed distinct clusters (note: protein length is not explicitly encoded in *SeqVec*). Also, these results indicated that the embeddings captured complex relationships between proteins which are not directly observable from sequence similarity alone as SCOPe was redundancy reduced at 40% sequence identity. Therefore, the new embeddings could complement sequence-based structural classification as it was shown that the sequence similarity does not necessarily lead to structural similarity [55].

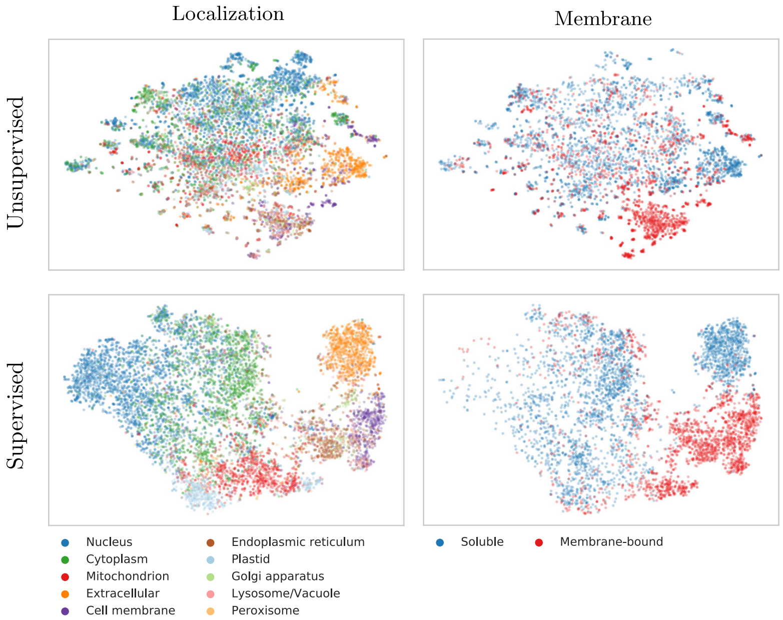

Fig. 2 t-SNE representations of SeqVec. Shown are t-SNE projections from embedded space onto a 2D representation; upper row: unsupervised 1024-dimensional "raw" ELMo-based SeqVec embeddings, averaged over all residues in a protein; lower row: supervised 32-dimensional ELMo-based SeqVec embeddings, reduced via per-protein machine learning predictions (data: redundancy reduced set from DeepLoc). Proteins were colored according to their localization (left column) or whether they are membrane-bound or water-soluble (right column). Left and right panel would be identical except for the color, however, on the right we had to leave out some points due to lacking membrane/non-membrane annotations. The upper row suggests that *SeqVec* embeddings capture aspects of proteins without ever seeing labels of localization or membrane, i.e. without supervised training. After supervised training (lower row), this information is transferred to, and further distilled by networks with simple architectures. After training, the power of SeqVeq embeddings to distinguish aspects of function and structure become even more pronounced, sometimes drastically so, as suggested by the almost fully separable clusters in the lower right panel

To further investigate the clusters emerging from the SCOPe data set, we colored the same data set based on protein functions (Fig. 3c) and kingdoms (Fig. 3d). This analysis revealed that many of the small, distinct clusters emerged based on protein functions. For instance, transferases and hydrolases formed many small clusters. When increasing the level of abstraction by coloring the proteins according to their kingdoms, we observed certain clusters to be dominated by e.g. eukaryotes. Comparing the different views captured in panels 3B-3D revealed connections, e.g. that all-beta or small proteins dominate in eukaryotes (compare blue and orange islands in Fig. 3b with the same islands in Fig. 3d – colored blue to mark eukaryotes).

CPU/GPU time used

Due to the sequential nature of LSTMs, the time required to embed a protein grows linearly with protein length. Depending on the available main memory or GPU memory, this process could be massively parallelized. To optimally use available memory, batches are typically based on tokens rather than on sentences. In order to retrieve embeddings, we sorted proteins according to their length and created batches of ≤15 K tokens that could still be handled by a single Nvidia GeForce GTX1080 with 8GB VRAM. The processing of a single protein took on average 0.027 s when applying this batch-strategy to the NetSurfP-2.0 data set (average protein length: 256 residues, i.e. shorter than proteins for which 3D structure is not known). The batch with the shortest proteins (on average 38 residues, corresponding to 15% of the average protein length in the whole data set) required about one tenth (0.003 s per protein, i.e. 11% of that for whole set). The batch containing the longest protein sequences in this data set (1578 residues on average, corresponding to 610% of average protein length in the whole data set), took about six times more (1.5 s per protein, i.e. 556% of that for whole set). When creating SeqVec for the DeepLoc set (average length: 558 residues; as this set does not require

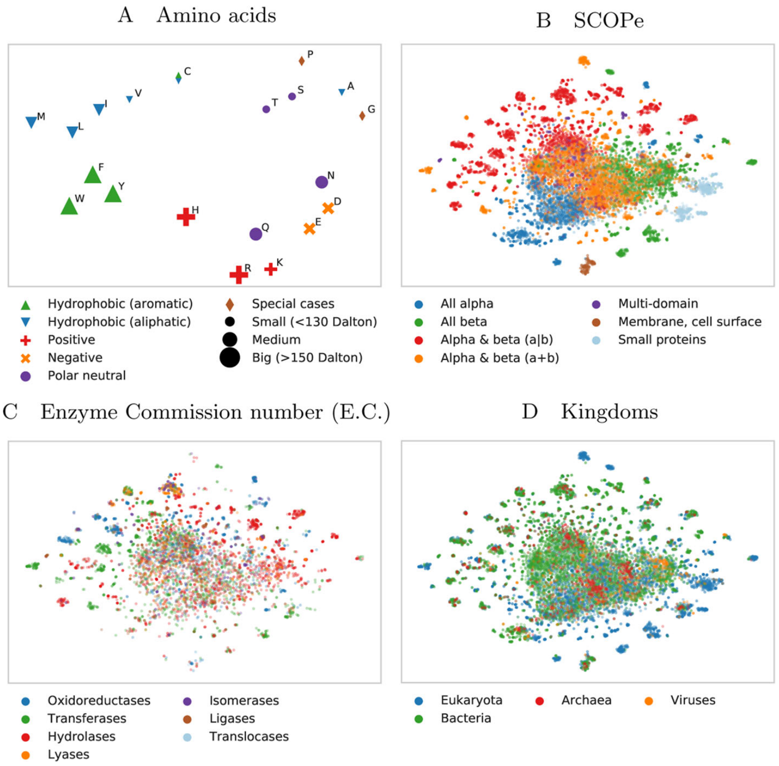

Fig. 3 Modeling aspects of the language of life. 2D t-SNE projections of unsupervised *SeqVec* embeddings highlight different realities of proteins and their constituent parts, amino acids. Panels **B** to **D** are based on the same data set (Structural Classification of Proteins – extended (SCOPe) 2.07, redundancy reduced at 40%). For these plots, only subsets of SCOPe containing proteins with the annotation of interest (enzymatic activity C and kingdom D) may be displayed. Panel **A**: the embedding space confirms: the 20 standard amino acids are clustered according to their biochemical and biophysical properties, i.e. hydrophobicity, charge or size. The unique role of Cysteine (C, mostly hydrophobic and polar) is conserved. Panel B: SeqVec embeddings capture structural information as annotated in the main classes in SCOPe without ever having been explicitly trained on structural features. Panel **C**: many small, local clusters share function as given by the main classes in the Enzyme Commission Number (E.C.). Panel **D**: similarly, small, local clusters represent different kingdoms of life

a 3D structure, it provides a more realistic view on the distribution of protein lengths), the average processing time for a single protein was 0.08 with a minimum of 0.006 for the batch containing the shortest sequences (67 residues on average) and a maximum of 14.5 s (9860 residues on average). On a single Intel i7–6700 CPU with 64GB RAM, processing time increased by roughly 50% to 0.41 s per protein, with a minimum and a maximum computation time of 0.06 and 15.3 s, respectively. Compared to an average processing time of one hour for 1000 proteins when using evolutionary information directly [46], this implied an average speed up of 120-fold on a single GeForce GTX1080 and 9-fold on a single i7–6700 when predicting structural features; the inference time of DeepSeqVec for a single protein is on average 0.0028 s.

Discussion

Transfer-learning alone not top

The context-dependent transfer-learning model ELMo [41] applied to proteins sequences (here dubbed *SeqVec*) clearly succeeded to model the language of protein sequences much better than simple schema (e.g. one-hot encoding), more advanced context-independent language models such as ProtVec (based on Word2vec [42, 43]), more advanced distillations of text-book knowledge (biophysical features used as input for prediction [2, 3]), and also some family-independent information about evolution as represented by the expertise condensed in the BLOSSUM62 matrix. In this sense, our approach worked. However, none of our SeqVec implementations reached today's best methods: NetSurfP-2.0 for secondary structure and protein disorder and DeepLoc for localization and membrane protein classification (Fig. 1, Table 1, Table 2). Clearly, “just” using SeqVec embeddings to train subsequent prediction methods did not suffice to crack the challenges. Due to computational limitations, testing models trained on larger sequence database, which may over-come this limitation, could not be tested. What about more advanced transfer-learning models, e.g. TransformerXL [56], or different pre-training objectives which model bidirectional contexts, e.g. Bert [57] or XLNet [58]? We have some evidence that transformer-based models might reach further (Elnaggar et al. in preparation), with competing groups already showing promising results [51]. Nevertheless, there is one major reality to remember: we model single protein sequences. Such models might learn the rules for “writing protein sequences” and still miss the constraints imposed by the “survival of the fittest”, i.e. by evolutionary selection.

On the other hand, some of our solutions appeared surprisingly competitive given the simplicity of the architectures. In particular, for the per-protein predictions, for which *SeqVec* clearly outperformed the previously popular *ProtVec* [42] approach and even commonly used expert solutions (Fig. 1, Table 2: no method tested other than the top-of-the-line *DeepLoc* reached higher numerical values). For that comparison, we used the same data sets but could not rigorously compare standard errors (SE) that were unavailable for other methods. Estimating standard errors for our methods suggested the differences to be statistically significant: > 7 SE throughout (exception: DeepLoc (Q10 = 78) and iLoc-Euk(Q10 = 68)). The results for localization prediction implied that frequently used methods using evolutionary information (all marked with shaded boxes in Table 2) did not clearly outperform our simple ELMo-based tool (DeepSeqVec-Loc in Table 2). This was very different for the per-residue prediction tasks: here almost all top methods using evolutionary information numerically outperformed the simple model built on the ELMo embeddings (DeepSeqVec in Fig. 1 and Table 1). However, all models introduced in this work were deliberately designed to be relatively simple to demonstrate the predictive power of *SeqVec*. More sophisticated architectures building up on *SeqVec* embeddings will likely outperform the approaches introduced here.

Combining SeqVec with evolutionary information for per-residue predictions still did not reach the top (set TS115: Q3(NetSurfP-2.0) = 85.3% vs. Q3(DeepProf + SeqVec) = 82.4%, Table 1). This might suggest some limit for the usefulness of the ELMo-based SeqVec embeddings. However, it might also point to the more advanced solutions realized by NetSurfP-2.0 which applies two LSTMs of similar complexity as our entire system (including ELMo) on top of their last step leading to 35 M (35 million) free parameters compared to about 244 K for DeepProf + SeqVec. Twenty times more free parameters might explain some fraction of the success. Due to limited GPU resources, we could not test how much.

Why did the ELMo-based approach improve more (relative to competition) for per-protein than for per-residue predictions? We can only speculate because none of the possible explanations have held consistently for all methods to which we have been applying ELMo embeddings over the recent six months (data not shown). For instance, the per-protein data sets were over two orders of magnitude smaller than those for per-residue predictions; simply because every protein constitutes one sample in the first and protein length samples for the second. SeqVec might have helped more for the smaller data sets because the unlabeled data is pre-processed so meaningful that less information needs to be learned by the ANN during per-protein prediction. This view was strongly supported by the t-SNE [53] results (Fig. 2, Fig. 3): ELMo apparently had learned the “grammar” of the language of life well enough to realize a very rough clustering of structural classes, protein function, localization and membrane/not. Another, yet complementary, explanation for this trend could be that the training of ELMo inherently provides a natural way of summarizing information of proteins of varying length. Other approaches usually learn this summarization step together with the actual prediction tasks which gets increasingly difficult the smaller the data set.

We picked four tasks as proof-of-principle for our ELMo/SeqVec approach. These tasks were picked because recent breakthroughs had been reported (e.g. NetSurfP-2.0 [46] and DeepLoc [47]) and those had made data for training and testing publicly available. We cannot imagine why our findings should not hold true for other tasks of protein prediction and invite the community to apply the *SeqVec* embeddings for their tasks. We assume the SeqVec embeddings to be more beneficial for small than for large data sets. For instance, we expect little or no gain in predicting inter-residue contacts, and more in predicting protein binding sites.

Good and fast predictions without using evolutionary information

Although our SeqVec embeddings were over five percentage points worse than the best method NetSurfP-2.0 (Table 1: TS115 Q3: 85.3 vs. 79.1), for some proteins (12% in CB513) DeepSeqVec performed better (Additional file 1: Figure S4). We expect those to be proteins with small or incorrect alignments, however, due to the fact that we did not have the alignments available used by NetSurfP-2.0, we could not quite establish the validity of this assumption (analyzing pre-computed alignments from ProteinNet [59] revealed no clear relation of the type: more evolutionary information leads to better prediction). However, the real strength of our solutions is its speed: SeqVec predicted secondary structure and protein disorder over 100-times faster (on a single 8GB GPU) than NetSurfP-2.0 when counting the time it needs to retrieve the evolutionary information summarized in alignment profiles although using the fastest available alignment method, namely MMseqs2 [36] which already can reach speed-up values of 100-times over PSI-BLAST [33]. For those who do not have enough resources for running MMSeqs2 and therefore have to rely on PSI-BLAST, the speed-up of our prediction becomes 10,000-fold. Even the 100-fold speed-up is so substantial that for some applications, the speedup might outweigh the reduction in performance. Embedding based approaches such as *SeqVec* suggest a promising solution toward solving one of the biggest challenges for computational biology: how to efficiently handle the exponentially increasing number of sequences in protein databases? Here, we showed that relevant information from large unannotated biological databases can be compressed into embeddings that condense and abstract the underlying biophysical principles. These embeddings, essentially the weights of a neural network, help as input to many problems for which smaller sets of annotated data are available (secondary structure, disorder, localization). Although the compression step needed to build the *SeqVec* model is very GPU-intensive, it can be performed in a centralized way using large clusters. After training, the model can be shipped and used on any consumer hardware. Such solutions are ideal to support researches without access to expensive cluster infrastructure.

Modeling the language of life?

SeqVec, our pre-trained ELMo adaption, learned to model a probability distribution over a protein sequence. The sum over this probability distribution constituted a very informative input vector for any machine learning task trying to predict protein features. It also picked up context-dependent protein motifs without explicitly explaining what these motifs are relevant for. In contrast, context-independent tools such as *ProtVec* [42] will always create the same vectors regardless of the residues surrounding this k-mer in a protein sequence.

Our hypothesis had been that the ELMo-based *SeqVec* embeddings trained on large databases of un-annotated protein sequences could extract a *probabilistic model of the language of life* in the sense that the resulting system will extract aspects relevant both for per-residue and per-protein prediction tasks. All results presented here have added independent evidence in full support of this hypothesis. For instance, the three state per-residue accuracy for secondary structure prediction improved by over eight percentage points through ELMo (Table 1, e.g. Q3: 79.1 vs. 70.3%), the per-residue MCC for protein disorder prediction also increased substantially (Table 1, e.g. MCC: 0.591 vs. 0.488). On the per-protein level, the improvement over the previously popular tool extracting “meaning” from proteins, *ProtVec*, was even more substantial (Table 1: e.g. Q10: 68% vs. 42%). We could demonstrate this reality even more directly using the t-SNE [53] results (Fig. 2 and Fig. 3): different levels of complexity ranging from single amino acids, over some localizations, structural features, functions and the classification of membrane/non-membrane had been implicitly learned by *SeqVec* without training. Clearly, our ELMo-driven implementation of transfer-learning fully succeeded to model some aspects of the language of life as proxied by protein sequences. How much more will be possible? Time will tell.

Conclusion

We have shown that it is possible to capture and transfer knowledge, e.g. biochemical or biophysical properties, from a large unlabeled data set of protein sequences to smaller, labelled data sets. In this first proof-of-principle, our comparably simple models have already reached promising performance for a variety of per-residue and per-protein prediction tasks obtainable from only single protein sequences as input, that is: without any direct evolutionary information, i.e. without profiles from multiple sequence alignments of protein families. This reduces the dependence on the time-consuming and computationally intensive calculation of protein profiles, allowing the prediction of per-residue and per-protein features of a whole proteome within less than an hour. For instance, on a single GeForce GTX 1080, the creation of embeddings and predictions of secondary structure and subcellular localization for the whole human proteome took about 32 min. Building more sophisticated architectures on top of *SeqVec* might increase sequence-based performance further.

Our new *SeqVec* embeddings may constitute an ideal starting point for many different applications in particular when labelled data are limited. The embeddings combined with evolutionary information might even improve over the best available methods, i.e. enable high-quality predictions. Alternatively, they might ease high-throughput predictions of whole proteomes when used as the only input feature. Alignment-free predictions bring speed and improvements for proteins for which alignments are not readily available or limited, such as for intrinsically disordered proteins, for the Dark Proteome, or for particular unique inventions of evolution. The trick was to tap into the potential of Deep

Learning through transfer learning from large repositories of unlabeled data by modeling the language of life.

Methods

Data

UniRef50 training of *SeqVec:* We trained ELMo on UniRef50 [32], a sequence redundancy-reduced subset of the UniProt database clustered at 50% pairwise sequence identity (PIDE). It contained 25 different letters (20 standard and 2 rare amino acids (U and O) plus 3 special cases describing either ambiguous (B, Z) or unknown amino acids (X); Additional file 1: Table S1) from 33 M proteins with 9,577,889,953 residues. In order to train ELMo, each protein was treated as a sentence and each amino acid was interpreted as a single word.

Visualization of embedding space: The current release of the “Structural Classification Of Proteins” (SCOPe, [54]) database (2.07) contains 14,323 proteins at a redundancy level of 40%. Functions encoded by the Enzyme Commission number (E.C., [60]) were retrieved via the “Structure Integration with Function, Taxonomy and Sequence” (SIFTS) mapping [61]. SIFTS allows, among other things, a residue-level mapping between UniProt and PDB entries and a mapping from PDB identifiers to E.C.s. If no function annotation was available for a protein or if the same PDB identifier was assigned to multiple E.C.s, it was removed from Fig. 3c. Taxonomic identifiers from UniProt were used to map proteins to one of the 3 kingdoms of life or to viruses. Again, proteins were removed if no such information was available. The number of iterations for the t-SNE projections was set again to 3000 and the perplexity was adjusted (perplexity = 5 for Fig. 3a and perplexity = 30 for Fig. 3b-d).

Per-residue level: secondary structure & intrinsic disorder (*NetSurfP-2.0*). To simplify comparability, we used the data set published with a recent method seemingly achieving the top performance of the day in secondary structure prediction, namely *NetSurfP-2.0* [46]. Performance values for the same data set exist also for other recent methods such as *Spider3* [62], *RaptorX* [63, 64] and *JPred4* [65]. The set contains 10,837 sequence-unique (at 25% PIDE) proteins of experimentally known 3D structures from the PDB [66] with a resolution of 2.5 Å (0.25 nm) or better, collected by the PISCES server [67]. DSSP [68] assigned secondary structure and intrinsically disordered residues are flagged (residues without atomic coordinates, i.e. REMARK-465 in the PDB file). The original seven DSSP states (+ 1 for unknown) were mapped upon three states using the common convention: [G,H, I] → H (helix), [B,E] → E (strand), all others to O (other; often misleadingly referred to as *coil* or *loop*). As the authors of NetSurfP-2.0 did not include the raw protein sequences in their public data set, we used the SIFTS file to obtain the original sequence. Only proteins with identical length in SIFTS and NetSurfP-2.0 were used. This filtering step removed 56 sequences from the training set and three from the test sets (see below: two from CB513, one from CASP12 and none from TS115). We randomly selected 536 (~ 5%) proteins for early stopping (*cross-training*), leaving 10,256 proteins for training. All published values referred to the following three test sets (also referred to as validation set): **TS115** [69]: 115 proteins from high-quality structures (< 3 Å) released after 2015 (and at most 30% PIDE to any protein of known structure in the PDB at the time); **CB513** [70]: 513 non-redundant sequences compiled 20 years ago (511 after SIFTS mapping); **CASP12** [71]: 21 proteins taken from the CASP12 free-modelling targets (20 after SIFTS mapping; all 21 fulfilled a stricter criterion toward non-redundancy than the two other sets; non-redundant with respect to all 3D structures known until May 2018 and all their relatives). Each of these sets covers different aspects of the secondary structure prediction problem: CB513 and TS115 only use structures determined by X-ray crystallography and apply similar cutoffs with respect to redundancy (30%) and resolution (2.5–3.0 Å). While these serve as a good proxy for a baseline performance, CASP12 might better reflect the true generalization capability for unseen proteins as it includes structures determined via NMR and Cryo-EM. Also, the strict redundancy reduction based on publication date reduces the bias towards well studied families. Nevertheless, toward our objective of establishing a proof-of-principle, these sets sufficed. All test sets had fewer than 25% PIDE to any protein used for training and cross-training (ascertained by the *NetSurfP-2.0* authors). To compare methods using evolutionary information and those using our new word embeddings, we took the *HHblits* profiles published along with the NetSurfP-2.0 data set.

Per-protein level: subcellular localization & membrane proteins (DeepLoc). Subcellular localization prediction was trained and evaluated using the *DeepLoc* data set [47] for which performance was measured for several methods, namely: LocTree2 [72], MultiLoc2 [73], SherLoc2 [74], CELLO [75], iLoc-Euk [52], WoLF PSORT [76] and YLoc [77]. The data set contained proteins from UniProtKB/Swiss-Prot [78] (release: 2016_04) with experimental annotation (code: ECO:0000269). The *DeepLoc* authors mapped these annotations to ten classes, removing all proteins with multiple annotations. All these proteins were also classified into *water-soluble* or *membrane-bound* (or as *unknown* if the annotation was ambiguous). The resulting 13,858 proteins were clustered through PSI-CD-HIT [79, 80] (version 4.0; at 30% PIDE or Eval< 10^{-6}). Adding the requirement that the alignment had to cover 80% of the shorter protein, yielded 8464 clusters. This set was split into training and testing by using the same proteins for testing as the authors of DeepLoc. The training set was randomly subdivided into 90% for training and 10% for determining early stopping (cross-training set).

Embedding terminology and related work

One-hot encoding (also known as *sparse encoding*) assigns each word (referred to as token in NLP) in the vocabulary an integer N used as the Nth component of a vector with the dimension of the vocabulary size (number of different words). Each component is binary, i.e. either 0 if the word is not present in a sentence/text or 1 if it is. This encoding drove the first application of machine learning that clearly improved over all other methods in protein prediction [1–3]. TF-IDF represents tokens as the product of "frequency of token in data set" times "inverse frequency of token in document". Thereby, rare tokens become more relevant than common words such as "the" (so called *stop words*). This concept resembles that of using k-mers for database searches [33], clustering [81], motifs [82, 83], and prediction methods [72, 76, 84–88]. Context-insensitive word embeddings replaced expert features, such as TF-IDF, by algorithms that extracted such knowledge automatically from unlabeled corpus such as Wikipedia, by either predicting the neighboring words, given the center word (skip-gram) or vice versa (CBOW). This became known in *Word2Vec* [43] and showcased for computational biology through *ProtVec* [43, 89]. ProtVec assumes that every token or word consists of three consecutive residues (amino acid 3-mers). During training, each protein sequence in *SwissProt* [78] is split into overlapping 3-mers and the skip-gram version of *word2vec* is used to predict adjacent 3-mers, given the 3-mer at the center. After training, protein sequences can be split into overlapping 3-mers which are mapped onto a 100-dimensional latent space. More specialized implementations are *mut2vec* [90] learning mutations in cancer, and *phoscontext2vec* [91] identifying phosphorylation sites. Even though the performance of context-insensitive approaches was pushed to its limits by adding sub-word information (FastText [92]) or global statistics on word co-occurance (GloVe [93]), their expressiveness remained limited because the models inherently assigned the same vector to the same word, regardless of its context. Context-sensitive word embeddings started a new wave of word embedding techniques for NLP in 2018: the embedding renders the meaning of words and phrases such as "*paper tiger*" dependent upon the context, allowing to account for the ambiguous meanings of words. Popular examples like ELMo [41] and Bert [57] have achieved state-of-the-art results in several NLP tasks. Both require substantial GPU computing power and time to be trained from scratch. One of the main differences between ELMo and Bert is their pre-training objective: while auto-regressive models like ELMo predict the next word in a sentence given all previous words, autoencoder-based models like Bert predict masked-out words given all words which were not masked out. However, in this work we focused on ELMo as it allows processing of sequences of variable length. The original ELMo model consists of a single, context-insensitive CharCNN [94] over the characters in a word and two layers of bidirectional LSTMs that introduce the context information of surrounding words (Fig. 4). The CharCNN transforms all characters within a single word via an embedding layer into vector space and runs multiple CNNs of varying window size (here: ranging from 1 to 7) and number of filters (here: 32, 64, …, 1024). In order to obtain a fixed-dimensional vector for each word, regardless of its length, the output of the CNNs is max-pooled and concatenated. This feature is crucial for NLP in order to be able to process words of variable length. As our words consist only of single amino acids, this layer learns an uncontextualized mapping of single amino acids onto a latent space. The first bi-directional LSTM operates directly on the output of the CharCNN, while the second LSTM layer takes the output of the first LSTM as input. Due to their sequential nature, the LSTM layers render the embeddings dependent on their context as their internal state always depends on the previous hidden state. However, the bidirectionality of the LSTMs would lead to information leakage, rendering the training objective trivial, i.e. the backward pass had already seen the word which needs to be predicted in the forward pass. This problem is solved by training the forward and the backward pass of the LSTMs independently, i.e. the forward pass is conditioned only on words to its left and vice versa. During inference the internal states of both directions are concatenated allowing the final embeddings to carry information from both sides of the context. As described in the original ELMo publication, the weights of the forward and the backward model are shared in order to reduce the memory overhead of the model and to combat overfitting. Even though, the risk of overfitting is small due to the high imbalance between number of trainable parameters (93 M) versus number of tokens (9.3B), dropout at a rate of 10% was used to reduce the risk of overfitting. This model is trained to predict the next amino acid given all previous amino acids in a protein sequence. To the best of our knowledge, the context-sensitive ELMo has not been adapted to protein sequences, yet.

ELMo adaptation

In order to adapt ELMo [41] to protein sequences, we used the standard ELMo configuration with the following changes: (i) reduction to 28 tokens (20 standard and

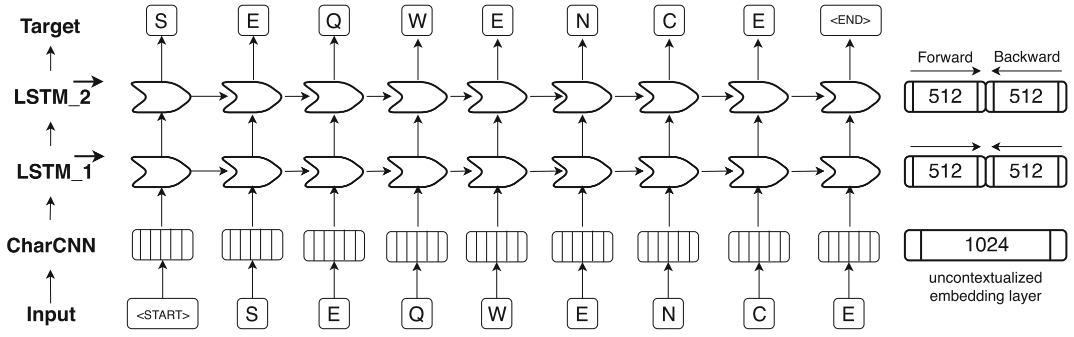

Fig. 4 ELMo-based architecture adopted for SeqVec. First, an input sequence, e.g. "S E Q W E N C E" (shown at bottom row), is padded with special tokens indicating the start ("<start>") and the end ("<end>") of the sentence (here: protein sequences). On the 2nd level (2nd row from bottom), character convolutions (CharCNN, [94]) map each word (here: amino acid) onto a fixed-length latent space (here: 1024-dimensional) without considering information from neighboring words. On the third level (3rd row from bottom), the output of the CharCNN-layer is used as input by a bidirectional Long Short Term Memory (LSTM, [45]) which introduces context-specific information by processing the sentence (protein sequence) sequentially. For simplicity, only the forward pass of the bi-directional LSTM-layer is shown (here: 512-dimensional). On the fourth level (4th row from bottom), the second LSTM-layer operates directly on the output of the first LSTM-layer and tries to predict the next word given all previous words in a sentence. The forward and backward pass are optimized independently during training in order to avoid information leakage between the two directions. During inference, the hidden states of the forward and backward pass of each LSTM-layer are concatenated to a 1024-dimensional embedding vector summarizing information from the left and the right context

2 rare (U,O) amino acids + 3 special tokens describing ambiguous (B,Z) or unknown (X) amino acids + 3 special tokens for ELMo indicating padded elements ('< MASK>') or the beginning ('<S>') or the end of a sequence ('</S>')), (ii) increase number of unroll steps to 100 to account for the increased length of protein sequences compared to sentences in natural languages, (iii) decrease number of negative samples to 20, (iv) increase token number to 9,577,889,953. After pre-training the ELMo architecture (1 CharCNN, 2 LSTM-Layers, see "Embedding terminology and related work" section and Fig. 4 for more details) with our parameters on UniRef50, the embedding model takes a protein sequence of arbitrary length and returns 3076 features for each residue in the sequence. These 3076 features were derived by concatenating the outputs of the three layers of ELMo, each describing a token with a vector of length 1024. The LSTM layers were composed of the embedding of the forward pass (first 512 dimensions) and the backward pass (last 512 dimensions). In order to demonstrate the general applicability of ELMo or *SeqVec* and to allow for easy integration into existing models, we neither fine-tuned the pre-trained model on a specific prediction task, nor optimized the combination of the three internal layers. Thus, researchers could just replace (or concatenate) their current machine learning inputs with our embeddings to boost their task-specific performance. Furthermore, it will simplify the development of custom models that fit other use-cases. For simplicity, we summed the components of the three 1024-dimensional vectors to form a single 1024-dimensional feature vector describing each residue in a protein.

Using SeqVec for predicting protein features

On the per-residue level, the predictive power of the new *SeqVec* embeddings was demonstrated by training a small two-layer Convolutional Neural Network (CNN) in PyTorch using a specific implementation [95] of the ADAM optimizer [96], cross-entropy loss, a learning rate of 0.001 and a batch size of 128 proteins. The first layer (in analogy to the sequence-to-structure network of earlier solutions [2, 3]) consisted of 32-filters each with a sliding window-size of w = 7. The second layer (structure-to-structure [2, 3]) created the final predictions by applying again a CNN (w = 7) over the output of the first layer. These two layers were connected through a rectified linear unit (ReLU) and a dropout layer [97] with a dropout-rate of 25% (Fig. 5, left panel). This simple architecture was trained independently on six different types of input, resulting in different number of free parameters. (i) DeepProf (14,000 = 14 k free parameters): Each residue was described by a vector of size 50 which included a one-hot encoding (20 features), the profiles of evolutionary information (20 features) from HHblits as published previously [46], the state transition probabilities of the Hidden-Markov-Model (7 features) and 3 features describing the local alignment diversity. (ii) DeepSeqVec (232 k free parameters): Each protein sequence was represented by the output of SeqVec. The

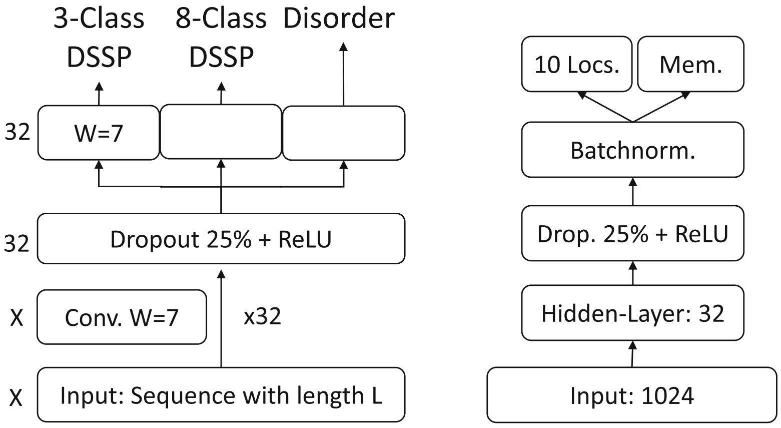

Fig. 5 Prediction tasks' architectures. On the left the architecture of the model used for the per-residue level predictions (secondary structure and disorder) is sketched, on the right that used for per-protein level predictions (localization and membrane/not membrane). The 'X', on the left, indicates that different input features corresponded to a difference in the number of input channels, e.g. 1024 for *SeqVec* or 50 for profile-based input. The letter 'W' refers to the window size of the corresponding convolutional layer (W = 7 implies a convolution of size 7 × 1)

resulting embedding described each residue as a 1024-dimensional vector. (iii) DeepProf+SeqVec (244 k free parameters): This model simply concatenated the input vectors used in (i) and (ii). (iv) DeepProtVec (25 k free parameters): Each sequence was split into overlapping 3-mers each represented by a 100-dimensional ProtVec [42]. (v) DeepOneHot (7 k free parameters): The 20 amino acids were encoded as one-hot vectors as described above. Rare amino acids were mapped to vectors with all components set to 0. Consequently, each protein residue was encoded as a 20-dimensional one-hot vector. (vi) DeepBLOSUM65 (8 k free parameters): Each protein residue was encoded by its BLOSUM65 substitution matrix [98]. In addition to the 20 standard amino acids, BLOSUM65 also contains substitution scores for the special cases B, Z (ambiguous) and X (unknown), resulting in a feature vector of length 23 for each residue.

On the per-protein level, a simple feed-forward neural network was used to demonstrate the power of the new embeddings. In order to ensure equal-sized input vectors for all proteins, we averaged over the 1024-dimensional embeddings of all residues in a given protein resulting in a 1024-dimensional vector representing any protein in the data set. ProtVec representations were derived the same way, resulting in a 100-dimensional vector. These vectors (either 100-or 1024 dimensional) were first compressed to 32 features, then dropout with a dropout rate of 25%, batch normalization [99] and a rectified linear Unit (ReLU) were applied before the final prediction (Fig. 5, right panel). In the following, we refer to the models trained on the two different input types as (i) DeepSeqVec-Loc (33 k free parameters): average over SeqVec embedding of a protein as described above and (ii) DeepProtVec-Loc (320 free parameters): average over ProtVec embedding of a protein. We used the following hyper-parameters: learning rate: 0.001, Adam optimizer with cross-entropy loss, batch size: 64. The losses of the individual tasks were summed before backpropagation. Due to the relatively small number of free parameters in our models, the training of all networks completed on a single Nvidia GeForce GTX1080 within a few minutes (11 s for DeepProtVec-Loc, 15 min for DeepSeqVec).

Evaluation measures

To simplify comparisons, we ported the evaluation measures from the publications we derived our data sets from, i.e. those used to develop *NetSurfP-2.0* [46] and *DeepLoc* [47]. All numbers reported constituted averages over all proteins in the final test sets. This work aimed at a proof-of-principle that the *SeqVec* embedding contain predictive information. In the absence of any claim for state-of-the-art performance, we did not calculate any significance values for the reported values.

Per-residue performance: Toward this end, we used the standard three-state per-residue accuracy (Q3 = percentage correctly predicted in either helix, strand, other [2]) along with its eight-state analog (Q8). Predictions of intrinsic disorder were evaluated through the Matthew's correlation coefficient (MCC [100]) and the False-Positive Rate (FPR) as those are more informative for tasks with high class imbalance. For completeness, we also provided the entire confusion matrices for both secondary structure prediction problems (Additional file 1: Figure S2). Standard errors were calculated over the distribution of each performance measure for all proteins.

Per-protein performance: The predictions whether a protein was membrane-bound or water-soluble were evaluated by calculating the two-state per set accuracy (Q2: percentage of proteins correctly predicted), and the MCC. A generalized MCC using the Gorodkin measure [101] for K (=10) categories as well as accuracy (Q10), was used to evaluate localization predictions. Standard errors were calculated using 1000 bootstrap samples, each chosen randomly by selecting a sub-set of the predicted test set that had the same size (draw with replacement).

Supplementary information

Supplementary information accompanies this paper at https://doi.org/10.1186/s12859-019-3220-8.

Additional file 1: Supporting online material (SOM) for: Modeling aspect of the language of life through transfer-learning protein sequences **Figure 1.** ELMo perplexity **Figure 2.** Confusion matrices for per-protein predictions using DeepSeqVec-Loc **Figure 3.** Confusion matrices for secondary structure predictions of DeepSeqVec **Figure 4.** Comparison of secondary structure prediction performance (Q3) between Netsurfp-2.0 and DeepSeqVec **Table S1.** Amino acid occurrences in UniRef50

Abbreviations

1D: One-dimensional – information representable in a string such as secondary structure or solvent accessibility; 3D structure: Three-dimensional coordinates of protein structure; 3D: Three-dimensional; ELMo: Embeddings from Language Models; MCC: Matthews-Correlation-Coefficient; MSA: Multiple sequence alignment; ProtVec: Context-independent embeddings from Word2vec-type approaches; Q10: Ten-state localization per-protein accuracy; Q3: Three-state secondary structure per-residue accuracy; Q8: Eight-state secondary structure per-residue accuracy; RSA: Relative solvent accessibility; SE: Standard error; SeqVec: embeddings introduced here, extracted by modeling un-annotated UniRef50 protein sequences with ELMo

Acknowledgements

The authors thank primarily Tim Karl for invaluable help with hardware and software and Inga Weise for support with many other aspects of this work. Last, not least, thanks to all those who deposit their experimental data in public databases, and to those who maintain these databases.

Authors contributions

AE and MH suggested to use ELMo for modeling protein sequences. AE adopted and trained ELMo. MH evaluated SeqVec embeddings on different data sets and tasks. YW helped with discussions about natural language processing. CD implemented the web-interface which allows to access and visualize the predictions and helped to improve the manuscript. DN helped with various problems regarding the code. FM and BR helped with the design of the experiment and to critically improve the manuscript. MH and AE drafted the manuscript and the other authors provided feedback. All authors read and approved the final manuscript.

Funding

This work was supported by a grant from the Alexander von Humboldt foundation through the German Ministry for Research and Education (BMBF: Bundesministerium fuer Bildung und Forschung) as well as by a grant from Deutsche Forschungsgemeinschaft (DFG–GZ: RO1320/4–1). We gratefully acknowledge the support of NVIDIA Corporation with the donation of two Titan GPU used for this research. We also want to thank the LRZ (Leibniz Rechenzentrum) for providing us access to DGX-V1.
The funding did not play any role in the design of the study, collection, analysis, and interpretation of data and in writing the manuscript.

Availability of data and materials

The pre-trained ELMo-based SeqVec model and a description on how to implement the embeddings into existing methods can be found here: https://github.com/Rostlab/SeqVec . Accessed 2nd May 2019.
Predictions on secondary structure, disorder and subcellular localization based on SeqVec can be accessed under: https://embed.protein.properties . Accessed 2nd May 2019.
The NetSurfP-2.0 data set [46] used for the evaluation of SeqVec on the task of secondary structure and disorder prediction are publicly available under: http://www.cbs.dtu.dk/services/NetSurfP/ . Accessed 2nd May 2019.
The DeepLoc data set [47] used for the evaluation of SeqVec on the task of subcellular localization prediction are publicly available under: http://www.cbs.dtu.dk/services/DeepLoc/data.php . Accessed 2nd May 2019.

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Informatics, Bioinformatics & Computational Biology - i12, TUM (Technical University of Munich), Boltzmannstr. 3, 85748 Garching/Munich, Germany. [2]TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany. [3]Leibniz Supercomputing Centre, Boltzmannstr. 1, 85748 Garching/Munich, Germany. [4]TUM Department of Informatics, Software Engineering and Business Information Systems, Boltzmannstr. 1, 85748 Garching/Munich, Germany. [5]Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, 85748 Garching/Munich, Germany. [6]TUM School of Life Sciences Weihenstephan (WZW), Alte Akademie 8, Freising, Germany. [7]Department of Biochemistry and Molecular Biophysics & New York Consortium on Membrane Protein Structure (NYCOMPS), Columbia University, 701 West, 168th Street, New York, NY 10032, USA.

Received: 3 May 2019 Accepted: 13 November 2019
Published online: 17 December 2019

References

1. Rost B, Sander C. Jury returns on structure prediction. Nat. 1992;360:540.
2. Rost B, Sander C. Prediction of protein secondary structure at better than 70% accuracy. J Mol Biol. 1993;232:584–99.
3. Rost B, Sander C. Improved prediction of protein secondary structure by use of sequence profiles and neural networks. Proc Natl Acad Sci. 1993;90:7558–62.
4. Barton GJ. Protein secondary structure prediction. Curr Opin Struct Biol. 1995;5:372–6.
5. Chandonia J-M, Karplus M. Neural networks for secondary structure and structural class predictions. Protein Sci. 1995;4:275–85.
6. Mehta PK, Heringa J, Argos P. A simple and fast approach to prediction of protein secondary structure from multiply aligned sequences with accuracy above 70%. Protein Sci. 1995;4:2517–25.
7. Rost B, Sander C. Combining evolutionary information and neural networks to predict protein secondary structure. Proteins Struct Funct Genet. 1994;19:55–72.
8. Solovyev VV, Salamov AA. Predicting a-helix and b-strand segments of globular proteins. Comput Appl Biol Sci. 1994;10:661–9.
9. Frishman D, Argos P. Knowledge-based protein secondary structure assignment. Proteins Struct Funct Genet. 1995;23:566–79.
10. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. J Mol Biol. 1999;292(2):195–202.
11. Bigelow H, Petrey D, Liu J, Przybylski D, Rost B. Predicting transmembrane beta-barrels in proteomes. Nucleic Acids Res. 2004;32:2566–77.
12. Rost B, Casadio R, Fariselli P. Topology prediction for helical transmembrane proteins at 86% accuracy. Protein Sci. 1996;5:1704–18.
13. Rost B, Casadio R, Fariselli P, Sander C. Transmembrane helix prediction at 95% accuracy. Protein Sci. 1995;4:521–33.

14. Rost B, Sander C. Conservation and prediction of solvent accessibility in protein families. Proteins Struct Funct Genet. 1994;20(3):216–26.
15. Radivojac P, Obradovic Z, Smith DK, Zhu G, Vucetic S, Brown CJ, Lawson JD, Dunker AK. Protein flexibility and intrinsic disorder. Protein Sci. 2004;13:71–80.
16. Schlessinger A, Rost B. Protein flexibility and rigidity predicted from sequence. Proteins. 2005;61(1):115–26.
17. Punta M, Rost B. PROFcon: novel prediction of long-range contacts. Bioinform. 2005;21(13):2960–8.
18. Peng K, Vucetic S, Radivojac P, Brown CJ, Dunker AK, Obradovic Z. Optimizing long intrinsic disorder predictors with protein evolutionary information. J Bioinforma Comput Biol. 2005;3(1):35–60.
19. Schlessinger A, Liu J, Rost B. Natively unstructured loops differ from other loops. PLoS Comput Biol. 2007;3(7):e140.
20. Schlessinger A, Punta M, Rost B. Natively unstructured regions in proteins identified from contact predictions. Bioinform. 2007;23(18):2376–84.
21. Nair R, Rost B. Better prediction of sub-cellular localization by combining evolutionary and structural information. Proteins. 2003;53(4):917–30.
22. Nair R, Rost B. Mimicking cellular sorting improves prediction of subcellular localization. J Mol Biol. 2005;348(1):85–100.
23. Marino Buslje C, Teppa E, Di Domenico T, Delfino JM, Nielsen M. Networks of high mutual information define the structural proximity of catalytic sites: implications for catalytic residue identification. PLoS Comput Biol. 2010; 6(11):e1000978.
24. Ofran Y, Rost B. Protein-protein interaction hot spots carved into sequences. PLoS Comput Biol. 2007;3(7):e119.
25. Ofran Y, Rost B. ISIS: interaction sites identified from sequence. Bioinform. 2007;23(2):e13–6.
26. Adzhubei IA, Schmidt S, Peshkin L, Ramensky VE, Gerasimova A, Bork P, Kondrashov AS, Sunyaev SR. A method and server for predicting damaging missense mutations. Nat Methods. 2010;7(4):248–9.
27. Bromberg Y, Rost B. SNAP: predict effect of non-synonymous polymorphisms on function. Nucleic Acids Res. 2007;35(11):3823–35.
28. Hayat S, Sander C, Marks DS, Elofsson A. All-atom 3D structure prediction of transmembrane β-barrel proteins from sequences. Proc Natl Acad Sci. 2015; 112(17):5413–8.
29. Marks DS, Colwell LJ, Sheridan R, Hopf TA, Pagnani A, Zecchina R, Sander C. Protein 3D structure computed from evolutionary sequence variation. PLoS One. 2011;6(12):e28766.
30. Marks DS, Hopf TA, Sander C. Protein structure prediction from sequence variation. Nat Biotechnol. 2012;30(11):1072.
31. Morcos F, Pagnani A, Lunt B, Bertolino A, Marks DS, Sander C, Zecchina R, Onuchic JN, Hwa T, Weigt M. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. Proc Natl Acad Sci. 2011;108(49):E1293–301.
32. Suzek BE, Wang Y, Huang H, McGarvey PB, Wu CH, UniProt C. UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. Bioinform. 2015;31(6):926–32.
33. Altschul SF, Madden TL, Schaeffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped Blast and PSI-Blast: a new generation of protein database search programs. Nucleic Acids Res. 1997;25:3389–402.
34. Remmert M, Biegert A, Hauser A, Soding J. HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment. Nat Methods. 2012;9(2):173–5.
35. Steinegger M, Meier M, Mirdita M, Vohringer H, Haunsberger SJ, Soding J. HH-suite3 for fast remote homology detection and deep protein annotation. BMC Bioinform. 2019;20(1):473.
36. Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nat Biotechnol. 2017; 35(11):1026.
37. Dunker AK, Babu MM, Barbar E, Blackledge M, Bondos SE, Dosztanyi Z, Dyson HJ, Forman-Kay J, Fuxreiter M, Gsponer J, et al. What's in a name? Why these proteins are intrinsically disordered. Intrinsically Disord Proteins. 2013;1(1):e24157.
38. Uversky VN, Radivojac P, Iakoucheva LM, Obradovic Z, Dunker AK. Prediction of intrinsic disorder and its use in functional proteomics. Methods Mol Biol. 2007;408:69–92.
39. Perdigao N, Heinrich J, Stolte C, Sabir KS, Buckley MJ, Tabor B, Signal B, Gloss BS, Hammang CJ, Rost B, et al. Unexpected features of the dark proteome. Proc Natl Acad Sci U S A. 2015.
40. Schafferhans A, O'Donoghue SI, Heinzinger M, Rost B. Dark proteins important for cellular function. Proteomics. 2018;18(21–22):1800227.
41. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L: Deep contextualized word representations. arXiv 2018,.https://arxiv.org/abs/1802.05365.
42. Asgari E, Mofrad MR. Continuous distributed representation of biological sequences for deep proteomics and genomics. PLoS One. 2015;10(11):e0141287.
43. Mikolov T, Chen K, Corrado G, Dean J: Efficient estimation of word representations in vector space. ArXiv 2013,https://arxiv.org/abs/1301.3781.
44. Schils E, Pd H. Characteristics of sentence length in running text. Literary Linguist Comput. 1993;8(1):20–6.
45. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80.
46. Klausen MS, Jespersen MC, Nielsen H, Jensen KK, Jurtz VI, Sonderby CK, Sommer MOA, Winther O, Nielsen M, Petersen B, et al. NetSurfP-2.0: Improved prediction of protein structural features by integrated deep learning. Proteins. 2019.
47. Almagro Armenteros JJ, Sonderby CK, Sonderby SK, Nielsen H, Winther O. DeepLoc: prediction of protein subcellular localization using deep learning. Bioinform. 2017;33(24):4049.
48. Anfinsen CB. Principles that govern the folding of protein chains. Sci. 1973; 181(4096):223–30.
49. Buchan DW, Jones DT. Improved protein contact predictions with the MetaPSICOV2 server in CASP12. Proteins. 2018;86:78–83.
50. Evans R, Jumper J, Kirkpatrick J, Sifre L, Green T, Qin C, Zidek A, Nelson A, Bridgland A, Penedones H. De novo structure prediction with deeplearning based scoring. Annu Rev Biochem. 2018;77(363–382):6.
51. Rives A, Goyal S, Meier J, Guo D, Ott M, Zitnick CL, Ma J, Fergus R. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. bioRxiv. 2019:622803.
52. Chou KC, Wu ZC, Xiao X. iLoc-Euk: a multi-label classifier for predicting the subcellular localization of singleplex and multiplex eukaryotic proteins. PLoS One. 2011;6(3):e18258.
53. Lvd M, Hinton G. Visualizing data using t-SNE. J Mach Learn Res. 2008; 9(Nov):2579–605.
54. Fox NK, Brenner SE, Chandonia J-M. SCOPe: structural classification of proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. Nucleic Acids Res. 2013;42(D1):D304–9.
55. Kosloff M, Kolodny R. Sequence-similar, structure-dissimilar protein pairs in the PDB. Proteins. 2008;71(2):891–902.
56. Dai Z, Yang Z, Yang Y, Cohen WW, Carbonell J, Le QV, Salakhutdinov R: Transformer-xl: Attentive language models beyond a fixed-length context. arXiv preprint arXiv:190102860 2019.
57. Devlin J, Chang M-W, Lee K, Toutanova K: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv: 181004805 2018.
58. Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov R, Le QV: XLNet: Generalized Autoregressive Pretraining for Language Understanding. arXiv preprint arXiv:190608237 2019.
59. AlQuraishi M. ProteinNet: a standardized data set for machine learning of protein structure. BMC Bioinform. 2019;20(1):311.
60. Bairoch A. The ENZYME database in 2000. Nucleic Acids Res. 2000;28(1):304–5.
61. Velankar S, Dana JM, Jacobsen J, Van Ginkel G, Gane PJ, Luo J, Oldfield TJ, O'donovan C, Martin M-J, Kleywegt GJ. SIFTS: structure integration with function, taxonomy and sequences resource. Nucleic Acids Res. 2012;41(D1):D483–9.
62. Heffernan R, Yang Y, Paliwal K, Zhou Y. Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility. Bioinform. 2017;33(18):2842–9.
63. Wang S, Li W, Liu S, Xu J. RaptorX-property: a web server for protein structure property prediction. Nucleic Acids Res. 2016;44(W1):W430–5.
64. Wang S, Peng J, Ma J, Xu J. Protein secondary structure prediction using deep convolutional neural fields. Sci Rep. 2016;6:18962.
65. Drozdetskiy A, Cole C, Procter J, Barton GJ. JPred4: a protein secondary structure prediction server. Nucleic Acids Res. 2015;43(W1):W389–94.
66. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE. The protein data bank. Nucleic Acids Res. 2000; 28(1):235–42.
67. Wang G, Dunbrack RL Jr. PISCES: a protein sequence culling server. Bioinform. 2003;19(12):1589–91.
68. Kabsch W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen bonded and geometrical features. Biopolym. 1983; 22:2577–637.

69. Yang Y, Gao J, Wang J, Heffernan R, Hanson J, Paliwal K, Zhou Y. Sixty-five years of the long march in protein secondary structure prediction: the final stretch? Brief Bioinform. 2016;19(3):482–94.
70. Cuff JA, Barton GJ. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. Proteins Struct Funct Genet. 1999;34(4):508–19.
71. Abriata LA, Tamò GE, Monastyrskyy B, Kryshtafovych A, Dal Peraro M. Assessment of hard target modeling in CASP12 reveals an emerging role of alignment-based contact prediction methods. Proteins. 2018;86:97–112.
72. Goldberg T, Hamp T, Rost B. LocTree2 predicts localization for all domains of life. Bioinform. 2012;28(18):i458–65.
73. Blum T, Briesemeister S, Kohlbacher O. MultiLoc2: integrating phylogeny and gene ontology terms improves subcellular protein localization prediction. BMC Bioinform. 2009;10:274.
74. Briesemeister S, Blum T, Brady S, Lam Y, Kohlbacher O, Shatkay H. SherLoc2: a high-accuracy hybrid method for predicting subcellular localization of proteins. J Proteome Res. 2009;8(11):5363–6.
75. Yu CS, Chen YC, Lu CH, Hwang JK. Prediction of protein subcellular localization. Proteins. 2006;64(3):643–51.
76. Horton P, Park KJ, Obayashi T, Fujita N, Harada H, Adams-Collier CJ, Nakai K. WoLF PSORT: protein localization predictor. Nucleic Acids Res. 2007;35(Web Server issue):W585–7.
77. Briesemeister S, Rahnenfuhrer J, Kohlbacher O. YLoc - an interpretable web server for predicting subcellular localization. Nucleic Acids Res. 2010; 38(Suppl):W497–502.
78. Boutet E, Lieberherr D, Tognolli M, Schneider M, Bansal P, Bridge AJ, Poux S, Bougueleret L, Xenarios I. UniProtKB/Swiss-Prot, the manually annotated section of the UniProt KnowledgeBase: how to use the entry view. Methods Mol Biol. 2016;1374:23–54.
79. Fu L, Niu B, Zhu Z, Wu S, Li W. CD-HIT: accelerated for clustering the next-generation sequencing data. Bioinform. 2012;28(23):3150–2.
80. Li W, Godzik A. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. Bioinform. 2006;22(13):1658–9.
81. Moussa M, Mandoiu II. Single cell RNA-seq data clustering using TF-IDF based methods. BMC Genomics. 2018;19(Suppl 6):569.
82. Bailey TL, Boden M, Buske FA, Frith M, Grant CE, Clementi L, Ren J, Li WW, Noble WS. MEME SUITE: tools for motif discovery and searching. Nucleic Acids Res. 2009;37(Web Server issue):W202–8.
83. Bernard G, Chan CX, Ragan MA. Alignment-free microbial phylogenomics under scenarios of sequence divergence, genome rearrangement and lateral genetic transfer. Sci Rep. 2016;6:28970.
84. Hamp T, Rost B. Evolutionary profiles improve protein-protein interaction prediction from sequence. Bioinform. 2015;31(12):1945–50.
85. Kuang R, Ie E, Wang K, Wang K, Siddiqi M, Freund Y, Leslie C. Profile-based string kernels for remote homology detection and motif extraction. J Bioinforma Comput Biol. 2005;3(3):527–50.
86. Leslie C, Eskin E, Weston J, Noble WS: Mismatch string kernels for SVM protein classification. Bioinform 2003:in press.
87. Nakai K, Horton P. PSORT: a program for detecting sorting signals in proteins and predicting their subcellular localization. Trends Biochem Sci. 1999;24(1):34–6.
88. Noble WS, Kuang R, Leslie C, Weston J. Identifying remote protein homologs by network propagation. FEBS J. 2005;272(20):5119–28.
89. Asgari E, McHardy AC, Mofrad MRK. Probabilistic variable-length segmentation of protein sequences for discriminative motif discovery (DiMotif) and sequence embedding (ProtVecX). Sci Rep. 2019;9(1):3577.
90. Kim S, Lee H, Kim K, Kang J. Mut2Vec: distributed representation of cancerous mutations. BMC Med Genet. 2018;11(2):33.
91. Xu Y, Song J, Wilson C, Whisstock JC. PhosContext2vec: a distributed representation of residue-level sequence contexts and its application to general and kinase-specific phosphorylation site prediction. Sci Rep. 2018;8.
92. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. Trans Assoc Comput Linguist. 2017;5:135–46.
93. Pennington J, Socher R, Manning C: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP): 2014. 1532–1543.
94. Kim Y, Jernite Y, Sontag D, Rush AM: Character-aware neural language models. In: Thirtieth AAAI Conference on Artificial Intelligence: 2016.
95. Reddi SJ, Kale S, Kumar S: On the convergence of adam and beyond. arXiv preprint arXiv:190409237 2019.
96. Kingma DP, Ba J: Adam: A method for stochastic optimization. arXiv preprint arXiv:14126980 2014.
97. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
98. Henikoff S, Henikoff JG. Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci. 1992;89(22):10915–9.
99. Ioffe S, Szegedy C: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:150203167 2015.
100. Matthews BW. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta. 1975;405:442–51.
101. Gorodkin J. Comparing two K-category assignments by a K-category correlation coefficient. Comput Biol Chem. 2004;28(5–6):367–74.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

A.1.2. Littmann, Heinzinger *et al.*, Nature Scientific Reports (2021)

www.nature.com/scientificreports

scientific reports

OPEN

Embeddings from deep learning transfer GO annotations beyond homology

Maria Littmann[1,2,6]✉, Michael Heinzinger[1,2,6], Christian Dallago[1,2], Tobias Olenyi[1] & Burkhard Rost[1,3,4,5]

Knowing protein function is crucial to advance molecular and medical biology, yet experimental function annotations through the Gene Ontology (GO) exist for fewer than 0.5% of all known proteins. Computational methods bridge this sequence-annotation gap typically through homology-based annotation transfer by identifying sequence-similar proteins with known function or through prediction methods using evolutionary information. Here, we propose predicting GO terms through annotation transfer based on proximity of proteins in the SeqVec embedding rather than in sequence space. These embeddings originate from deep learned language models (LMs) for protein sequences (SeqVec) transferring the knowledge gained from predicting the next amino acid in 33 million protein sequences. Replicating the conditions of CAFA3, our method reaches an F_{max} of 37 ± 2%, 50 ± 3%, and 57 ± 2% for BPO, MFO, and CCO, respectively. Numerically, this appears close to the top ten CAFA3 methods. When restricting the annotation transfer to proteins with < 20% pairwise sequence identity to the query, performance drops (F_{max} BPO 33 ± 2%, MFO 43 ± 3%, CCO 53 ± 2%); this still outperforms naïve sequence-based transfer. Preliminary results from CAFA4 appear to confirm these findings. Overall, this new concept is likely to change the annotation of proteins, in particular for proteins from smaller families or proteins with intrinsically disordered regions.

Abbreviations

BERT	Bidirectional Encoder Representations from Transformers (particular deep learning language model)
BP(O)	Biological process (ontology) from GO
CAFA	Critical Assessment of Functional Annotation
CC(O)	Cellular component (ontology) from GO
ELMo	Embeddings from Language Models
GO	Gene ontology
GOA	Gene Ontology Annotation
k-NN	K-nearest neighbor
LK	Limited-knowledge
LM	Language model
LSTMs	Long-short-term-memory cells
M	Million
MF(O)	Molecular function (ontology) from GO
NK	No-knowledge
PIDE	Percentage pairwise sequence identity
RI	Reliability index
RMSD	Root-mean-square deviation

[1]Department of Informatics, Bioinformatics and Computational Biology, i12, TUM (Technical University of Munich), Boltzmannstr. 3, Garching, 85748 Munich, Germany. [2]TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany. [3]Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, Garching, 85748 Munich, Germany. [4]School of Life Sciences Weihenstephan (TUM-WZW), TUM (Technical University of Munich), Alte Akademie 8, Freising, Germany. [5]Department of Biochemistry and Molecular Biophysics, Columbia University, 701 West, 168th Street, New York, NY 10032, USA. [6]These authors contributed equally: Maria Littmann and Michael Heinzinger. ✉email: littmann@rostlab.org

GO captures cell function through hierarchical ontologies. All organisms rely on the correct functioning of their cellular workhorses, namely their proteins involved in almost all roles, ranging from molecular functions (MF) such as chemical catalysis of enzymes to biological processes or pathways (BP), e.g., realized through signal transduction. Only the perfectly orchestrated interplay between proteins allows cells to perform more complex functions, e.g., the aerobic production of energy via the citric acid cycle requires the interconnection of eight different enzymes with some of them being multi-enzyme complexes[1]. The Gene Ontology (GO)[2] thrives to capture this complexity and to standardize the vocabulary used to describe protein function in a human- and machine-readable manner. GO separates different aspects of function into three hierarchies: MFO (Molecular Function Ontology), BPO (biological process ontology), and CCO, i.e. the cellular component(s) or subcellular localization(s) in which the protein acts.

Computational methods bridge the sequence-annotation gap. As the experimental determination of complete GO numbers is challenging, the gap between the number of proteins with experimentally verified GO numbers and those with known sequence but unknown function (sequence-annotation gap) remains substantial. For instance, UniRef100[3] (UniProt[3] clustered at 100% percentage pairwise sequence identity, PIDE) contains roughly 220 M (million) protein sequences of which fewer than 1 M have annotations verified by experts (Swiss-Prot[3] evidence codes EXP, IDA, IPI, IMP, IGI, IEP, TAS, or IC).

Computational biology has been bridging the sequence-annotation gap for decades[4–11], based on two different concepts: (1) sequence similarity-based transfer (or *homology-based inference*) which copies the annotation from one protein to another if that is sequence similar enough because proteins of similar sequence have similar function[12]. In more formal terms: given a query Q of unknown and a template T of known function (F_t): IF PIDE(Q,T) > threshold θ, transfer annotation F_t to Q. (2) *De-novo* methods predict protein function through machine learning[5]. If applicable, the first approach tends to out-perform the second[13–16] although it largely misses discoveries[17]. The progress of computational methods has been monitored by CAFA (*Critical Assessment of Functional Annotation*)[9,18,19], an international collaboration for advancing and assessing methods that bridge the sequence-annotation gap. CAFA takes place every 2–3 years with its fourth instance (CAFA4) currently being evaluated.

Here, we introduce a novel approach transferring annotations using the similarity of embeddings from language models (LMs: SeqVec[20] and ProtBert[21]) rather than the similarity of sequence. Using embedding space proximity has helped information retrieval in natural language processing (NLP)[22–25]. By learning to predict the next amino acid given the entire previous sequence on unlabeled data (only sequences without any phenotype/label), e.g., SeqVec learned to extract features describing proteins useful as input to different tasks (transfer learning). Instead of transferring annotations from the labeled protein T with the highest percentage pairwise sequence identity (PIDE) to the query Q, we chose T as the protein with the smallest distance in embedding space ($\mathrm{DIST}^{\mathrm{emb}}$) to Q. This distance also proxied the reliability of the prediction serving as threshold above which hits are considered too distant to infer annotations. Instead of picking the top hit, annotations can be inferred from the *k* closest proteins where *k* has to be optimized. In addition, we evaluate the influence of the type of LM used (SeqVec[20] vs. ProtBert[21]). Although the LMs were never trained on GO terms, we hypothesize LM embeddings to implicitly encode information relevant for the transfer of annotations, i.e., capturing aspects of protein function because embeddings have been shown to capture rudimentary features of protein structure and function[20,21,26,27].

Results and discussion

Simple embedding-based transfer almost as good as CAFA3 top-10. First, we predicted GO terms for all 3328 CAFA3 targets using the Gene Ontology Annotation (GOA) data set *GOA2017* (Methods), removed all entries identical to CAFA3 targets (PIDE = 100%; set: *GOA2017-100*) and transferred the annotations of the closest hit (k = 1; closest by Euclidean distance) in this set to the query. When applying the *NK evaluation mode* (no-knowledge available for query, Methods/CAFA3), the embedding-transfer reached $F_{\max}$ scores (Eq. 3) of 37 ± 2% for BPO (precision: P = 39 ± 2%, recall: R = 36 ± 2%, Eqs. 1/2), F1 = 50 ± 3% for MFO (P = 54 ± 3%, R = 47 ± 3%), and F1 = 57 ± 2% for CCO (P = 61 ± 3%, R = 54 ± 3%; Table 1, Fig. 1, Fig. S1). Errors were estimated through the 95% confidence intervals (± 1.96 stderr). Replacing the Euclidean by cosine distance (more standard amongst those working with embeddings, e.g., in NLP) changed little (Table 1; for simplicity, we only used Euclidean from here on). In the sense that the database with annotations to transfer (GOA2017) had been available before the CAFA3 submission deadline (February 2017), our predictions were directly comparable to CAFA3[19]. This embedding-based annotation transfer clearly outperformed the two CAFA3 baselines (Fig. 1: the simple *BLAST* for sequence-based annotation transfer, and the *Naïve* method assigning GO terms statistically based on database frequencies, here *GOA2017*); it would have performed close to the top ten CAFA3 competitors (in particular for BPO: Fig. 1) had the method competed at CAFA3.

Performance did not change when replacing global average with maximum pooling (Table 1). While averaging over long proteins could lead to information loss in the resulting embeddings, we did not observe a correlation between performance and protein length (Fig. S2, Table S1). In order to obtain the embeddings, we processed query and lookup protein the same way. If those have similar function and similar length, their embeddings might have lost information in the same way. This loss might have "averaged out" to generate similar embeddings.

Including more neighbors ($k > 1$) only slightly affected $F_{\max}$ (Table S2; all $F_{\max}$ averages for $k = 2$ to $k = 10$ remained within the 95% confidence interval of that for $k = 1$). When taking all predictions into account independent of a threshold in prediction strength referred to as the reliability index (RI, Methods; i.e., even low confidence annotations are transferred), the number of predicted GO terms increased with higher *k* (Table S3). The average number of GO terms annotated per protein in *GOA2017* already reached 37, 14, 9 for BPO, MFO, CCO, respectively. When including all predictions independent of their strength (RI) our method predicted more

Data set	*Embeddings*	F_{max}		
		BPO	*MFO*	*CCO*
GOA2017	*SeqVec*	37 ± 2%	50 ± 3%	57 ± 2%
	SeqVec (Cosine)	37 ± 2%	50 ± 3%	58 ± 2%
	SeqVec (maximum pooling)	35 ± 2%	52 ± 3%	58 ± 2%
	ProtBert	36 ± 2%	49 ± 3%	59 ± 2%
	BLAST	26%	42%	46%
GOA2017X	*SeqVec*	31 ± 2%	51 ± 3%	56 ± 2%
GOA2020	*SeqVec*	51 ± 2%	61 ± 3%	65 ± 2%
	ProtBert	50 ± 2%	59 ± 2%	65 ± 2%
	BLAST	31%	53%	58%

Table 1. Performance for CAFA3 targets for simple GO annotation transfers*. *Mean F_{max} values for GO term predictions using embeddings from two different language models (*SeqVec* or *ProtBert*) or sequence similarity (*BLAST*) for the data sets *GOA2017-100* (2017), *GOA2017X* (2017), and *GOA2020-100* (2020) used for annotation transfer (note: the notation '-100′ implies that any entry in the data set with PIDE = 100% to any CAFA3 protein had been removed). By default, embedding distance was assessed by Euclidean distance (Eq. 4; exception marked *cosine*), and per-residue embeddings were pooled by global average pooling (exception marked maximum pooling). All values were compiled for picking the single top hit (k = 1) and using the CAFA3 targets from the NK and full evaluation mode[19]. For all simple annotation transfers (embedding- and sequence-based), performance was higher for the more recent data sets (*GOA2020* vs. *GOA2017*). Error estimates are given as 95% confidence intervals. F_{max} values were computed using the CAFA3 tool[18,19].

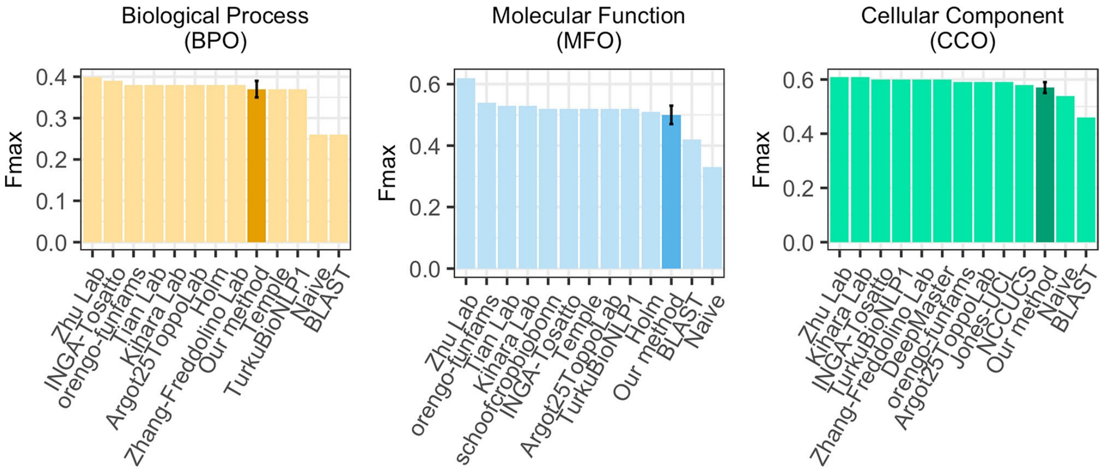

Figure 1. **F_{max} for simplest embedding-based transfer (k = 1) and CAFA3 competitors.** Using the data sets and conditions from CAFA3, we compared the F_{max} of the simplest implementation of the embedding-based annotation transfer, namely the greedy (k = 1) solution in which the transfer comes from exactly one closest database protein (dark bar) for the three ontologies (BPO, MFO, CCO) to the top ten methods that—in contrast to our method—did compete at CAFA3 and to two background approaches "BLAST" (homology-based inference) and "Naïve" (assignment of terms based on term frequency) (lighter bars). The result shown holds for the *NK* evaluation mode (no knowledge), i.e., only using proteins that were novel in the sense that they had no prior annotations. If we had developed our method before CAFA3, it would have almost reached the tenth place for MFO and CCO, and ranked even slightly better for BPO. Error bars (for our method) marked the 95% confidence intervals.

terms for CCO and BPO than expected from this distribution even for $k = 1$. Only for MFO the average (11.7 terms) predicted was slightly lower than expected for $k = 1$ (number of terms exploded for $k > 1$: Table S3). While precision dropped with adding terms, recall increased (Table S3). To avoid overprediction and given that k hardly affected F_{max}, we chose $k = 1$. This choice might not be best in practice: considering more than one hit ($k > 1$) might help when the closest hit only contains unspecific terms. However, such a distinction will be left to expert users.

When applying the *LK evaluation mode* (limited-knowledge, i.e., query already has some annotation about function, Methods/CAFA3), the embedding-based annotation transfer reached F_{max} scores of 49 ± 1%, 52 ± 2%,

	PIDE	d_{SeqVec}	$d_{ProtBert}$
PIDE	1	0.293	0.248
d_{SeqVec}		1	0.576
$d_{ProtBert}$			1

Table 2. Embedding and sequence similarity correlated*. **PIDE* percentage pairwise sequence identity, d_{SeqVec} similarity in SeqVec[20] embeddings (Eq. 5); $d_{ProtBert}$ similarity in ProtBert[21] embeddings (Eq. 5). All values represent Spearman's correlation coefficients calculated for 434,001 sequence pairs. For all pairs, the significance was *p*-value < 2.2e−16, i.e., significant at the level of the precision of the software R[28]. The similarity between sequence and embeddings correlated less than the two different types of embeddings, namely SeqVec and ProtBert with each other. In order to highlight the trivial symmetry of the matrix, only the upper diagonal was given.

and 58 ± 3% for BPO, MFO, and CCO, respectively (Fig. S3). Thus, the embedding-based annotation transfer reached higher values for proteins with prior annotations (LK evaluation mode) than for novel proteins without any annotations (NK evaluation mode; Table 1); the same was true for the CAFA3 top-10 for which the F_{max} scores increased even more than for our method for BPO and MFO, and less for CCO (Fig. 1, Fig. S3). In the *LK* mode, predictions are evaluated for proteins for which 1–2 GO ontologies had annotations while those for another ontology (or two) were added after the CAFA3 deadline[9,19]. While supervised training uses such labels; our approach did not since we had excluded all CAFA3 targets explicitly from the annotation transfer database (*GOA2017*). Thus, our method could not benefit from previous annotations, i.e., *LK* and *NK* should be identical. The observed differences were most likely explained by how F_{max} is computed. The higher F_{max} score, especially for BPO, might be explained by data set differences, e.g. LK average number of BPO annotations was 19 compared to 26 for NK. Other methods might have reached even higher by training on known annotations.

Embedding-based transfer successfully identified distant relatives. Embeddings condense information learned from sequences; identical sequences produce identical embeddings: if PIDE(Q,T) = 100%, then $DIST^{emb}(Q,T) = 0$ (Eq. 4). We had assumed a simple relation: the more similar two sequences, the more similar their embeddings because the underlying LMs only use sequences as input. Nevertheless, we observed embedding-based annotation transfer to outperform (higher F_{max}) sequence-based transfer (Table 1, Fig. 1). This suggested embeddings to capture information beyond raw sequences. Explicitly calculating the correlation between sequence and embedding similarity for 431,224 sequence pairs from CAFA3/*GOA2017*-100, we observed a correlation of $\rho = 0.29$ (Spearman's correlation coefficient, *p*-value < 2.2e−16; Table 2). Thus, sequence and embedding similarity correlated at an unexpectedly low level. However, our results demonstrated that embedding similarity identified more distant relatives than sequence similarity (Figs. S1, S4).

In order to quantify how different embeddings for proteins Q and T can still share GO terms, we redundancy reduced the *GOA2017* database used for annotation transfers at distance thresholds of decreasing PIDE with respect to the queries (in nine steps from 90 to 20%, Table S6). By construction, all proteins in *GOA2017-100* had PIDE < 100% to all CAFA3 queries (Q). If the pair (Q,T) with the most similar embedding was also similar in sequence, embedding-based would equal sequence-based transfer. At lower PIDE thresholds, e.g., PIDE < 20%, reliable annotation transfer through simple pairwise sequence alignment is no longer possible[14,29–32]. Although embeddings-based transfer tended to be slightly less successful for pairs with lower PIDE (Fig. 2: bars decrease toward right), the drop appeared small; on top, at almost all PIDE values, embedding-transfer remained above BLAST, i.e., sequence-based transfer (Fig. 2: most bars higher than reddish line – error bars show 95% confidence intervals). The exception was for MFO at PIDE < 30% and PIDE < 20% for which the F_{max} scores from sequence-based transfer (BLAST) were within the 95% confidence interval (Fig. 2). This clearly showed that our approach benefited from information available through embeddings but not through sequences, and that at least some protein pairs close in embedding and distant in sequence space might function alike. In order to correctly predict the next token, protein LMs have to learn complex correlations between residues as it is impossible to remember the multitude of all possible amino acid combinations in hundreds of millions to billions of protein sequences. This forces models to abstract higher level features from sequences. For instance, secondary structure can directly be extracted from embeddings through linear projection[26]. The LMs (SeqVec & ProtBert) might even have learned to find correlations between protein pairs diverged into the "midnight zone" sequence comparison in which sequence similarity becomes random[29,33]. Those cases are especially difficult to detect by the most successful search methods such as BLAST[34] or MMseqs2[35] relying on finding similar seeds missing at such diverged levels.

Ultimately, we failed to really explain why the abstracted level of sequences captured in embeddings outperformed raw sequences. One attempt at addressing this question led to displaying cases for which one of the two worked better (Fig. S5). Looking in more detail at outliers (embeddings more similar than sequences), we observed that embedding-based inference tended to identify more reasonable hits in terms of lineage or structure. For instance, for the uncharacterized transporter YIL166C (UniProt identifier P40445) from *Saccharomyces cerevisiae* (baker's yeast), the closest hit in SeqVec embedding space was the high-affinity nicotinic acid transporter (P53322) also from *Saccharomyces cerevisiae.* Both proteins belong to the allantoate permease family while the most sequence-similar hit (with PIDE = 31%) was the gustatory and odorant receptor 22 (Q7PMG3) from the insect *Anopheles gambiae* belonging to the gustatory receptor family. Experimental 3D structures were

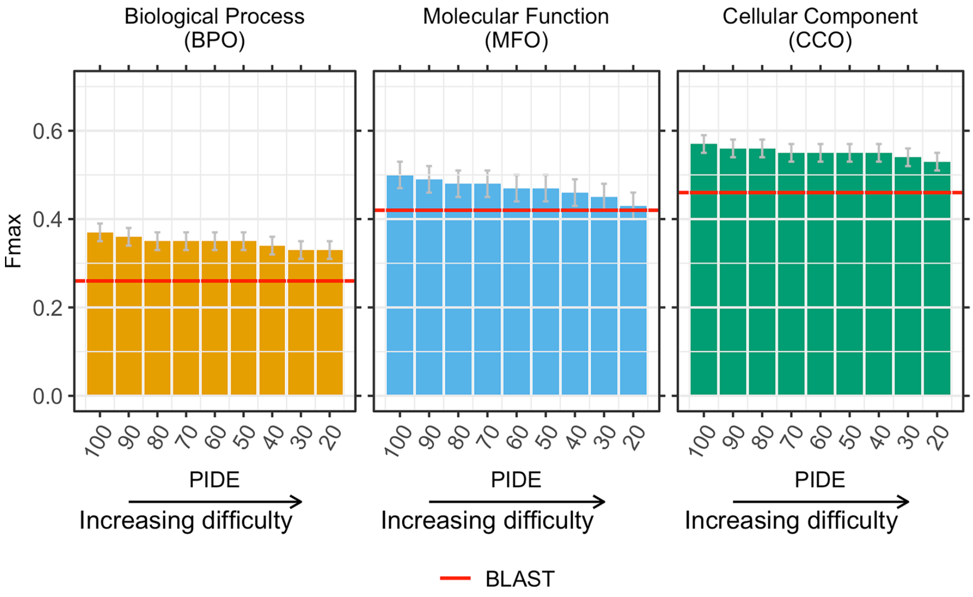

Figure 2. Embedding-based transfer succeeded for rather diverged proteins. After establishing the low correlation between embedding- and sequence-similarity, we tested how the level of percentage pairwise sequence identity (PIDE, x-axes) between the query (protein without known GO terms) and the transfer database (proteins with known GO terms, here subsets of *GOA2017*) affected the performance of the embedding-based transfer. Technically, we achieved this by removing proteins above a certain threshold in PIDE (decreasing toward right) from the transfer database. The y-axes showed the F_{max} score as compiled by CAFA3[19]. If embedding similarity and sequence identity correlated, our method would fall to the level of the reddish lines marked by BLAST. On the other hand, if the two were completely uncorrelated, the bars describing embedding-transfer would all have the same height (at least within the standard errors marked by the gray vertical lines at the upper end of each bar), i.e., embedding-based transfer would be completely independent of the sequence similarity between query and template (protein of known function). The observation that all bars tended to fall toward the right implied that embedding and sequence identity correlated (although for CCO, F_{max} remained within the 95% confidence interval of F_{max} for *GOA2017-100*). The observation that our method remained mostly above the baseline predictions demonstrates that embeddings capture important orthogonal information. Error bars indicate 95% confidence intervals.

not available for any of the three proteins. However, comparative modeling using Swiss-Model[36] revealed that both the target and the hit based on SeqVec were mapped to the same template (root-mean-square deviation (RMSD) = 0.3 Å) (Fig. S6a) while the hit based on sequence similarity was linked to a different structure (with RMSD = 16.8 Å) (Fig. S6b). Similarly, for the GDSL esterase/lipase At3g48460 (Q9STM6) from *Arabidopsis thaliana*, the closest hit in ProtBert embedding space was the GDSL esterase/lipase 2 (Q9SYF0) also from *Arabidopsis thaliana* while the most sequence-similar hit was the UDP-glucose 4-epimerase (Q564Q1) from *Caenorhabditis elegans*. The target and the embedding-based hit are both hydrolases belonging to the same CATH superfamily while the sequence-based hit is an isomerase and not annotated to any CATH superfamily. Comparative modeling suggested similar structures for target and embedding hit (RMSD = 2.9 Å) (Fig. S6c) while the structure found for the sequence-based hit similarity was very different (RMSD = 26 Å) (Fig. S6d). This suggested embeddings to capture structural features better than just raw sequences. Homology-based inference depends on many parameters that can especially affect the resulting sequence alignment for distantly related proteins. Possibly, embeddings are more robust in identifying those more distant evolutionary relatives.

Embedding-based transfer benefited from non-experimental annotations.

Unlike the data set made available by CAFA3, annotations in our *GOA2017* data set were not limited to experimentally verified annotations. Instead, they included annotations inferred by computational biology, homology-based inference, or by "author statement evidence", i.e., through information from publications. Using *GOA2017X,* the subset of *GOA2017-100* containing only experimental terms, our method reached F_{max} = 31 ± 2% (P = 28 ± 2%, R = 34 ± 2%), 51 ± 3% (P = 53 ± 3%, R = 49 ± 3%), and 56 ± 2% (P = 55 ± 3%, R = 57 ± 3%) for BPO, MFO, and CCO, respectively. Compared to using *GOA2017-100*, the performance dropped significantly for BPO (F_{max} = 37 ± 2% for *GOA2017-100,* Table 1); it decreased slightly (within 95% confidence interval) for CCO (F_{max} = 57 ± 2% for *GOA2017-100,* Table 1); and it increased slightly (within 95% confidence interval) for MFO (F_{max} = 50 ± 3% for *GOA2017-100*, Table 1). Thus, less reliable annotations might still help, in particular for BPO. Annotations for

BPO may rely more on information available from publications that is not as easily quantifiable experimentally as annotations for MFO or CCO.

Many of the non-experimental annotations constituted sequence-based annotation transfers. Thus, non-experimental annotations might have helped because they constituted an implicit merger of sequence and embedding transfer. Adding homologous proteins might "bridge" sequence and embedding space by populating embedding space using annotations transferred from sequence space. The weak correlation between both spaces supported this speculation because protein pairs with very similar sequences may differ in their embeddings and vice versa.

Improving annotations from 2017 to 2020 increased performance significantly. For CAFA3 comparisons, we only used data available before the CAFA3 submission deadline. When running new queries, annotations will be transferred from the latest GOA. We used *GOA2020-100* (from 02/2020 removing the CAFA3 targets) to assess how the improvement of annotations from 2017 to 2020 influenced annotation transfer (Table 1). On *GOA2020-100,* SeqVec embedding-based transfer achieved $F_{max} = 50 \pm 2\%$ ($P = 50 \pm 3\%$, $R = 50 \pm 3\%$), $60 \pm 3\%$ ($P = 52 \pm 3\%$, $R = 71 \pm 3\%$), and $65 \pm 2\%$ ($P = 57 \pm 3\%$, $R = 75 \pm 3\%$) for BPO, MFO, and CCO, respectively, for the *NK* evaluation mode (Table 1). This constituted a substantial increase over *GOA2017-100* (Table 1).

The large performance boost between *GOA2017* and *GOA2020* suggested the addition of many relevant GO annotations. However, for increasingly diverged pairs (Q,T), we observed a much larger drop in F_{max} than for *GOA2017* (Fig. 2, Fig. S4). In the extreme, *GOA2020-20* (PIDE(Q,T) < 20%) with $F_{max} = 33 \pm 2\%$ (BPO), $44 \pm 2\%$ (MFO), and $54 \pm 2\%$ (CCO) fell to the same level as *GOA2017-20* (Figs. 2, S4). These results suggested that many of the relevant GO annotations were added for proteins sequence-similar to those with existing annotations. Put differently, many helpful new experiments simply refined previous computational predictions.

Running BLAST against *GOA2020-100* for sequence-based transfer (choosing the hit with the highest PIDE) showed that sequence-transfer also profited from improved annotations (difference in F_{max} values for BLAST in Table 1). However, while F_{max} scores for embedding-based transfer increased the most for BPO, those for sequence-based transfer increased most for MFO. Embedding-transfer still outperformed BLAST for the *GOA2020-100* set (Fig. S4c).

Even when constraining annotation transfer to sequence-distant pairs, our method outperformed BLAST against *GOA2020-100* in terms of F_{max} at least for BPO and for higher levels of PIDE in MFO/CCO (Fig. S4c). However, comparing the results for BLAST on the *GOA2020-100* set with the performance of our method for subsets of very diverged sequences (e.g. PIDE < 40% for *GOA2020-40*) under-estimated the differences between sequence- and embedding-based transfer, because the two approaches transferred annotations from different data sets. For a more realistic comparison, we re-ran BLAST only considering hits below certain PIDE thresholds (for comparability we could not do this for CAFA3). As expected, performance for BLAST decreased with PIDE (Fig. S4 lighter bars), e.g., for PIDE < 20%, F_{max} fell to 8% for BPO, 10% for MFO, and 11% for CCO (Fig. S4c lighter bars) largely due to low coverage, i.e., most queries had no hit to transfer annotations from. At this level (and for the same set), the embedding-based transfer proposed here, still achieved values of $33 \pm 2\%$ (BPO), $44 \pm 2\%$ (MFO), and $54 \pm 2\%$ (CCO). Thus, our method made reasonable predictions at levels of sequence identity for which homology-based inference (BLAST) failed completely.

Performance confirmed by new proteins. Our method and especially the threshold to transfer a GO term were "optimized" using the CAFA3 targets. Without any changes in the method, we tested a new data set of 298 proteins, *GOA2020-new*, with proteins for which experimental GO annotations have been added since the CAFA4 submission deadline (02/2020; Method). Using the thresholds optimized for CAFA3 targets (0.35 for BPO, 0.28 for MFO, 0.29 for CCO, Fig. 3), our method reached $F_1 = 50 \pm 11\%$, $54 \pm 5\%$, and $66 \pm 8\%$ for BPO, MFO, and CCO, respectively. For BPO and CCO, the performance was similar to that for the CAFA3 targets; for MFO it was slightly below but within the 95% CI (Table 1). For yet a different set, submitted for MFO to CAFA4, the first preliminary evaluation published during ISMB2020[37], also suggested our approach to make it to the top-ten, in line with the *post facto* CAFA3 results presented here.

Embedding similarity influenced performance. Homology-based inference works best for pairs with high PIDE. Analogously, we assumed embedding-transfer to be best for pairs with high embedding similarity, i.e., low Euclidean distance (Eq. 4). We used this to define a reliability index (RI, Eq. 5). For the *GOA2020-100* set, the minimal RI was 0.24. The CAFA evaluation determined 0.35 for BPO, 0.28 for MFO, and 0.29 for CCO as thresholds leading to optimal performance as measured by F_{max} (Fig. 3 dashed grey lines marked these thresholds). For all ontologies, precision and recall were almost constant for lower RIs (up to ~ 0.3). For higher RIs, precision increased, and recall decreased as expected (Fig. 3). While precision increased up to 82% for BPO, 91% for MFO, and 70% for CCO, it also fluctuated for high RIs (Fig. 3). This trend was probably caused by the low number of terms predicted at these RIs. For CCO, the RI essentially did not correlate with precision. This might point to a problem in assessing annotations for which the trivial Naïve method reached values of $F_{max} \sim 55\%$ outperforming most methods. Possibly, some prediction of the type "organelle" is all that is needed to achieve a high F_{max} in this ontology.

Similar performance for different embeddings. We compared embeddings derived from two different language models (LMs). So far, we used embeddings from SeqVec[20]. Recently, ProtBert, a transformer-based approach using a masked language model objective (Bert[38]) instead of auto-regression and more protein sequences (BFD[39,40]) during pre-training, was shown to improve secondary structure prediction[21]. Replacing

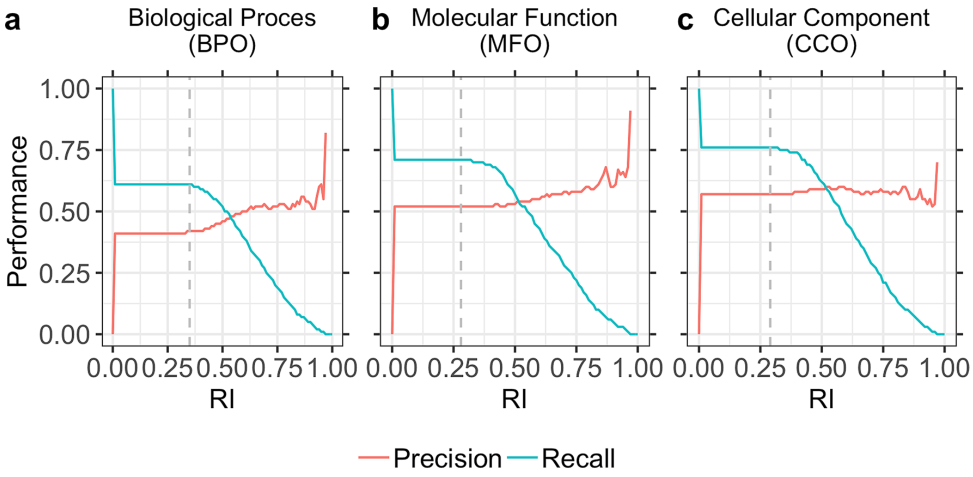

Figure 3. Precision and recall for different reliability indices (RIs). We defined a reliability index (RI) measuring the strength of a prediction (Eq. 5), i.e., for the embedding proximity. Precision (Eq. 1) and recall (Eq. 2) were almost constant for lower RIs (up to ~ 0.3) for all three ontologies (BPO, MFO, CCO). For higher RIs, precision increased while recall dropped. However, due to the low number of terms predicted at very high RIs (> 0.8), precision fluctuated and was not fully correlated with RI. Panel (**a**) shows precision and recall for BPO, panel (**b**) for MFO, and panel (**c**) for CCO. Dashed vertical lines marked the thresholds used by the CAFA3 tool to compute F_{max}: 0.35 for BPO, 0.28 for MFO, and 0.29 for CCO. At least for BPO and MFO higher RI values correlated with higher precision, i.e., users could use the RI to estimate how good the predictions are likely to be for their query, or to simply scan only those predictions more likely to be correct (e.g. RI > 0.8).

SeqVec by ProtBert embeddings to transfer annotations, our approach achieved similar F_{max} scores (Table 1). In fact, the ProtBert F_{max} scores remained within the 95% confidence intervals of those for SeqVec (Table 1). Similar results were observed when using *GOA2017-100* (Table 1).

On the one hand, the similar performance for both embeddings might indicate that both LMs extracted equally beneficial aspects of function, irrespective of the underlying architecture (LSTMs in SeqVec, transformer encoders in ProtBert) or training set (SeqVec used UniRef50 with ~ 33 M proteins, ProtBert used BFD with ~ 2.1B proteins). On the other hand, the similar F_{max} scores might also highlight that important information was lost when averaging over the entire protein to render fixed-size vectors. The similarity in F_{max} scores was less surprising given the high correlation between SeqVec and ProtBert embeddings (ρ = 0.58, p-value < 2.2e-16; Table 2). The two LMs correlated more with each other than either with PIDE (Table 2).

No gain from simple combination of embedding- and sequence-based transfer.

All three approaches toward annotation transfer (embeddings from SeqVec or ProtBert, and sequence) had strengths; although performing worse on average, for some proteins sequence-transfer performed better. In fact, analyzing the pairs for which embedding-based transfer or sequence-based transfer outperformed the other method by at least four percentage points ($|F_{max}(BLAST)-F_{max}(embeddings)| \geq 4$) illustrated the expected cases for which PIDE was high and embedding similarity low, and vice versa, along with more surprising cases for which low PIDE still yielded better predictions than relatively high embedding RIs (Fig. S5). Overall, these results (Fig. S5) again underlined that LM embeddings abstract information from sequence that are relevant for comparisons and not captured by sequences alone. However, it also indicates that even protein pairs with low embedding similarity can share similar GO terms. In fact, embedding similarity for SeqVec embeddings only weakly correlated with GO term similarity (Spearman rank coefficient ρ = 0.28, *p*-value < 2.2e−16), but proteins with identical GO annotations were on average more likely to be close than proteins with more different GO annotations (Fig. S7). The similarity of GO terms for two proteins was proxied through the Jaccard index (Eq. 7). More details are provided in the SOM.

To benefit from the cases where BLAST outperformed our approach, we tried simple combinations: firstly, we considered all terms predicted by embeddings from either SeqVec or ProtBert. Secondly, reliability scores were combined leading to higher reliability for terms predicted in both approaches than for terms only predicted by one. None of those two improved performance (Table S4, method SeqVec/ProtBert). Other simple combinations also failed so far (Table S4, method SeqVec/ProtBert/BLAST). Future work might improve performance through more advanced combinations.

Case study: embedding-based annotation transfer for three proteomes.

Due to its simplicity and speed, embedding-based annotation transfer can easily be applied to novel proteins to shed light on their potential functionality. We applied our method to the proteomes of three different proteomes: human (20,370 proteins from Swiss-Prot) as a well-researched proteome, the fungus *Armillaria ostoyae* (22,192 proteins, 0.01%

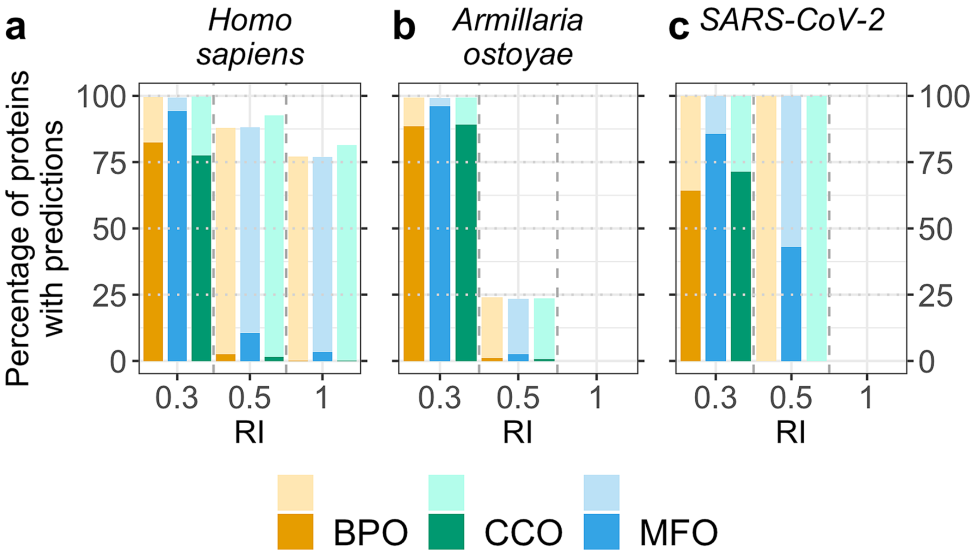

Figure 4. Fraction of proteomes with predicted GO terms. We applied our method to three proteomes (animal: *Homo sapiens*, fungus: *Armillaria ostoyae*, and virus: SARS-Cov-2) and monitored the fraction of proteins in each proteome for which our method predicted GO terms for different thresholds in embedding similarity (RI, Eq. 5). We show predictions for RI = 1.0 ("self-hits"), RI = 0.5 (with an expected precision/recall = 0.5), and RI = 0.3 (CAFA3 thresholds). Darker colored bars indicate predictions using *GOA2020X* as lookup set (only experimentally verified GO annotations) and lighter colors indicate predictions using *GOA2020* as lookup set (using all annotations in GOA). (**a**) The human proteome is well-studied (all 20,370 proteins are in Swiss-Prot) and for most proteins, GO annotations are available, but those annotations are largely not experimentally verified (very small, dark-colored bars vs large, lighter-colored bars at RI = 1.0). (**b**) The proteome of the fungus *Armillaria ostoyae* appears more exceptional (0.01% of the 22,192 proteins were in Swiss-Prot); at RI ≥ 0.5, predictions could be made only for 25% of the proteins when also using unverified annotations and none of the proteins already had any GO annotations. (**c**) While annotations were unknown for most proteins of the novel virus SARS-CoV-2 (no coverage at RI = 1), many annotations could be transferred from the human SARS coronavirus (SARS-CoV) and the bat coronavirus HKU3 (BtCoV) allowing GO term predictions for all proteins at reliability values ≥ 0.5.

of these in Swiss-Prot), as one of the oldest (2500 years) and largest ($4*10^5$ kg/spanning over 10 km^2) living organisms known today[41], and SARS-CoV-2, the virus causing COVID-19 (14 proteins). At RI = 1.0, annotations were inferred from proteins of this organism ("self-hits"). Using only experimentally verified annotations (lookup data set *GOA2020X*), revealed both how few proteins were directly annotated (self-hits) in these organisms and how much of the sequence-annotation gap is gapped through embedding-based inference (Fig. 4: bars with darker orange, blue, green for BPO, CCO, and MFO respectively). In particular, for self-hits, i.e., proteins with 100% pairwise sequence identity (PIDE) to the protein with known annotation, it became obvious how few proteins in human have explicit experimental annotation (sum over all around 270), while through embedding-based inference up to 80% of all human proteins could be annotated through proteins from other organisms (light bars in Fig. 4 give results for the entire *GOA2020* which is dominated by annotations not directly verified by experiment). For the other two proteomes from the fungus (*Armillaria ostoyae*) and the coronavirus (SARS-CoV-2), there were no inferences at this high level. On the other end of including all inferences as assessed through the data presented in all other figures and tables (i.e., at the default thresholds), for all three proteins most proteins could be annotated directly from experimentally verified annotations through embeddings (three left-most bars in Fig. 4 for BPO, CCO, and MFO). In fact, when including all GO annotations from GOA (lookup set *GOA2020*), almost all proteins in all three proteomes could be annotated (Fig. 4: lighter colored left-most bars close to fraction of 1, i.e., all proteins). For SARS-CoV-2, our method reached 100% coverage (prediction for all proteins) already at RI≥0.5 (Fig. 4c, lighter colors, middle bars) through well-studied, similar viruses such as the human SARS coronavirus (SARS-CoV). RI = 0.5 represent roughly a precision and recall of 50% for all three ontologies (Fig. 3). For *Armillaria ostoyae*, almost no protein was annotated through self-hits even when using unverified annotations (Fig. 4b: no bar at RI = 1). At RI = 0.5, about 25% of the proteins were annotated.

Case study: embedding-based annotation transfer for SARS-CoV-2 proteome. Given the relevance of SARS-CoV-2, we did not only apply our method to predict GO terms (BPO, MFO, and CCO) for all 14 SARS-CoV-2 proteins (taken from UniProt[3]; all raw predictions were made available as additional files named predictions_$emb_$ont.txt replacing the variables $emb and $ont as follows: $emb = seqvec|protbert, and $ont = bpo|mfo|cco), but also investigated the resulting annotations further. While the two replicase polyproteins pp1a and pp1ab can also be split further into up to 12 non-structural proteins resulting in 28 proteins[42], we used the definition from UniProt identifying 14 different proteins.

Step 1: confirmation of known annotations. Out of the 42 predictions (14 proteins in 3 ontologies), 12 were based on annotation transfers using proteins from the human SARS coronavirus (SARS-CoV), and 13 on proteins from the bat coronavirus HKU3 (BtCoV). CCO predictions appeared reasonable with predicted locations mainly associated with the virus (e.g., viral envelope, virion membrane) or the host (e.g., host cell Golgi apparatus, host cell membrane). Similarly, MFO predictions often matched well-known annotations, e.g., the replicase polyproteins 1a and 1ab were predicted to be involved in RNA-binding as confirmed by UniProt. In fact, annotations in BPO were known for 7 proteins (in total 40 GO terms), in MFO for 6 proteins (30 GO terms), and in CCO for 12 proteins (68 GO terms). Only three of these annotations were experimentally verified. With our method, we predicted 25 out of the 40 GO terms for BPO (63%), 14/30 for MFO (47%), and 59/68 for CCO (87%). Even more annotations were similar to the known GO annotations but were more or less specific (Table S5 summarized all predicted and annotated GO leaf terms, the corresponding names can be found in the additional files predictions_$emb_$ont.txt).

Step 2: new predictions. Since the GO term predictions matched well-characterized proteins, predictions might provide insights into the function of proteins without or with fewer annotations. For example, function and structure of the non-structural protein 7b (Uniprot identifier P0DTD8) are not known except for a transmembrane region of which the existence was supported by the predicted CCO annotation "*integral component of the membrane*" and "*host cell membrane*". This CCO annotation was also correctly predicted by the embedding-based transfer from an *Ashbya gossypii* protein. Additionally, we predicted "*transport of virus in host, cell to cell*" for BPO and "*proton transmembrane transporter activity*" for MFO. This suggested non-structural protein 7b to play a role in transporting the virion through the membrane into the host cell. Visualizing the leaf term predictions in the GO hierarchy could help to better understand very specific annotations. For the BPO annotation of the non-structural protein 7b, the tree revealed that this functionality constituted two major aspects: The interaction with the host and the actual transport to the host (Fig. S10). To visualize the predicted terms in the GO hierarchy, for example the tool NaviGO[43] can be used which can help to interpret the GO predictions given for the SARS-CoV-2 proteins here.

Comparing annotation transfers based on embeddings from SeqVec and from ProtBert showed that 16 of the 42 predictions agreed for the two different language models (LMs). For five predictions, one of the two LMs yielded more specific annotations, e.g., for the nucleoprotein (Uniprot identifier P0DTC9) which is involved in viral genome packaging and regulation of viral transcription and replication. For this protein, SeqVec embeddings found no meaningful result, while ProtBert embeddings predicted terms such as "*RNA polymerase II preinitiation complex*" and "*positive regulation of transcription by RNA polymerase II*" fitting to the known function of the nucleoprotein. This example demonstrated how the combination of results from predictions using different LMs may refine GO term predictions.

Conclusions

We introduce a new concept for the prediction of GO terms, namely the annotation transfer based on similarity of embeddings obtained from deep learning language models (LMs). This approach conceptually replaces sequence information by complex embeddings that capture some non-local information beyond sequence similarity. The underlying LMs (SeqVec & ProtBert) are highly involved and complex, and their training is time-consuming and data intensive. Once that is done, those pre-trained LMs can be applied, their abstracted understanding of the language of life as captured by protein sequences can be transferred to yield an extremely simple, yet effective novel method for annotation transfer. This novel prediction method complements homology-based inference. Despite its simplicity, this new method outperformed by several margins of statistically significance homology-based inference ("BLAST") with F_{max} values of BPO + **11** ± 2% (F_{max}(embedding)-F_{max}(sequence)), MFO + **8** ± 3%, and CCO + **11** ± 2% (Table 1, Fig. 1); it even might have reached the top ten, had it participated at CAFA3 (Fig. 1). Embedding-based transfer remained above the average for sequence-based transfer even for protein pairs with PIDE < 20% (Fig. 2), i.e., embedding similarity worked for proteins that diverged beyond the recognition in pairwise alignments (Figs. S2 & S3). Embedding-based transfer is also blazingly fast to compute, i.e., around 0.05 s per protein. The only time-consuming step is computing embeddings for all proteins in the lookup database which needs to be done only once; it took about 30 min for the entire human proteome. GO annotations added from 2017 to 2020 improved both sequence- and embedding-based annotation transfer significantly (Table 1). Another aspect of the simplicity is that, at least in the context of the CAFA3 evaluation, the choice of none of the two free parameters really mattered: embeddings from both LMs tested performed, on average, equally, and the number of best hits (k-nearest neighbors) did not matter much (Table S2). The power of this new concept is generated by the degree to which embeddings implicitly capture important information relevant for protein structure and function prediction. One reason for the success of our new concept was the limited correlation between embeddings and sequence (Table 2). Additionally, the abstraction of sequence information in embeddings appeared to make crucially meaningful information readily available (Fig. S6). This implies that embeddings have the potential to revolutionize the way sequence comparisons are carried out.

Methods

Generating embedding space. The embedding-based annotation transfer introduced here requires each protein to be represented by a fixed-length vector, i.e., a vector with the same dimension for a protein of 30 and another of 40,000 residues (maximal sequence length for ProtBert). To this end, we used SeqVec[20] to represent each protein in our data set by a fixed size embedding. SeqVec is based on ELMo[44] using a stack of LSTMs[45] for auto-regressive pre-training[46,47] i.e., predicting the next token (originally a word in a sentence, here an amino acid in a protein sequence), given all previous tokens. Two independent stacks of LSTMs process the sequence from both directions. During pre-training, the two directions are joined by summing their losses; concatenating the hidden states of both directions during inference lets supervised tasks capture bi-directional context. For SeqVec, three layers, i.e., one uncontextualized CharCNN[48] and two bi-directional LSTMs, were trained on each protein in UniRef50 (UniProt[3] clustered at 50% PIDE resulting in ~ 33 M proteins). In order to increase regularization, the weights of the token representation (CharCNN) as well as the final Softmax layer were shared between the two LSTM directions, and a 10% dropout rate was applied. For SeqVec, the CharCNN as well as each LSTM has a hidden state of size 512, resulting in a total of 93 M free parameters. As only unlabeled data (no phenotypical data) was used (self-supervised training), the embeddings could not capture any explicit information such as GO numbers. Thus, SeqVec does not need to be retrained for subsequent prediction tasks using the embeddings as input. The hidden states of the pre-trained model are used to extract features. Corresponding to its hidden state size, SeqVec outputs for each layer and each direction a 512-dimensional vector; in this work, only the forward and backward passes of the first LSTM layer were extracted and concatenated into a matrix of size L * 1024 for a protein with L residues. While the auto-regressive pre-training only allowed to gather contextual information from either direction, the concatenation of the representations allowed our approach to benefit from bi-directional context. A fixed-size representation was then derived by averaging over the length dimension, resulting in a vector of size 1024 for each protein (Fig. S11). This simple way of information pooling (also called *global average pooling*) outperformed in many cases more sophisticated methods in NLP[49] and showed competitive performance in bioinformatics for some tasks[20,21,26]. Based on experience from NLP[49,50], we also investigated the effect of using a different pooling strategy, i.e., maximum pooling, to derive fixed size representations from SeqVec embeddings.

To evaluate the effect of using different LMs to generate the embeddings, we also used a transformer-based LM trained on protein sequences (ProtBert-BFD[21], here simply referred to as *ProtBert*). ProtBert is based on the LM BERT[38] (Bidirectional Encoder Representations from Transformers[51]) which processes sequential data through the self-attention mechanism[52]. Self-attention compares all tokens in a sequence to all others in parallel, thereby capturing long-range dependencies better than LSTMs. BERT also replaced ELMo's auto-regressive objective by masked language modeling during pre-training, i.e., reconstructing corrupted tokens from the input, which enables to capture bi-directional context. ProtBert was trained with 30 attention layers, each having 16 attention heads with a hidden state size of 1024 resulting in a total of 420 M free parameters which were optimized on 2.1B protein sequences (BFD)[39,40] which is 70 times larger than UniRef50. The output of the last attention layer of ProtBert was used to derive a 1024-dimensional embedding for each residue. As for SeqVec, the resulting L * 1024 matrix was pooled by averaging over protein length providing a fixed-size vector of dimension 1024 for each protein. Usually, BERT's special CLS-token is used for sequence-classification tasks[38] as it is already optimized during pre-training on summarizing sequence information by predicting whether two sentences are consecutive in a document or not. In the absence of such a concept for proteins, this second loss was dropped from ProtBert's pre-training rendering the CLS token without further fine-tuning on supervised tasks uninformative.

Embeddings derived from LMs change upon retraining the model with a different random seed, even using the same data and hyper-parameters. They are likely to change more substantially when switching the training data or tuning hyper-parameters. As retraining LMs is computationally (and environmentally) expensive, we leave assessing the impact of fine-tuning LMs to the future.

Generating the embeddings for all human proteins using both SeqVec and ProtBert allowed estimating the time required for the generation of the input to our new method. Using a single Nvidia GeForce GTX1080 with 8 GB vRAM and dynamic batching (depending on the sequence length), this took, on average, about 0.05 s per protein[21].

Data set. To create a database for annotation transfer, we extracted protein sequences with annotated GO terms from the Gene Ontology Annotation (GOA) database[53–55] (containing 29,904,266 proteins from UniProtKB[3] in February 2020). In order to focus on proteins known to exist, we only extracted records from Swiss-Prot[56]. Proteins annotated only at the ontology roots, i.e. proteins limited to "GO:0003674" (molecular_function), "GO:0008150" (biological_process), or "GO:0005575" (cellular_component) were considered meaningless and were excluded. The final data set *GOA2020* contained 295,558 proteins (with unique identifiers, IDs) described by 31,485 different GO terms. The GO term annotation for each protein includes all annotated terms and all their parent terms. Thereby, proteins are, on average, annotated by 37 terms in BPO, 14 in MFO, and 9 in CCO. Counting only leaves brought the averages to 3 in BPO, 2 in MFO, and 3 in CCO.

For comparison to methods that contributed to CAFA3[19], we added another data set *GOA2017* using the GOA version available at the submission deadline of CAFA3 (Jan 17, 2017). After processing (as for *GOA2020*), *GOA2017* contained 307,287 proteins (unique IDs) described by 30,124 different GO terms. While we could not find a definite explanation for having fewer proteins in the newer database (*GOA2020* 295 K proteins vs. *GOA2017* with 307 K), we assume that it originated from major changes in GO including the removal of obsolete and inaccurate annotations and the refactoring of MFO[2].

The above filters neither excluded GOA annotations inferred from phylogenetic evidence and author statements nor those based on computational analysis. We constructed an additional data set, *GOA2017X* exclusively containing proteins annotated in Swiss-Prot as experimental (evidence codes EXP, IDA, IPI, IMP, IGI, IEP, TAS, or IC) following the CAFA3 definition[19]. We further excluded all entries with PIDE = 100% to any CAFA3 target bringing *GOA2017X* to 303,984 proteins with 28,677 different GO terms.

Performance evaluation. The targets from the CAFA3 challenge[19] were used to evaluate the performance of our new method. Of the 130,827 targets originally released for CAFA3, experimental GO annotations were obtained for 3328 proteins at the point of the final benchmark collection in November 2017[19]. This set consisted of the following subsets with experimental annotations in each sub-hierarchy of GO: BPO 2145, MFO 1101, and CCO 1097 (more details about the data set are given in the original CAFA3 publication[19]).

We used an additional data set, dubbed *GOA2020-new*, containing proteins added to GOA after February 2020, i.e., the point of accession for the GOA set used during the development of our method in preparation for CAFA4. This set consisted of 298 proteins with experimentally verified GO annotations and without any identical hits (i.e. 100% PIDE) in the lookup set *GOA2020*.

In order to expand the comparison of the transfer based on sequence- and embedding similarity, we also reduced the redundancy through applying CD-HIT and PSI-CD-HIT[57] to the *GOA2020* and *GOA2017* sets against the evaluation set at thresholds θ of PIDE = 100, 90, 80, 70, 60, 50, 40, 30 and 20% (Table S6 in the Supporting Online Material (SOM) for more details about these nine subsets).

We evaluated our method against two baseline methods used at CAFA3, namely *Naïve* and *BLAST*, as well as, against CAFA3′s top ten[19]. We computed standard performance measures. True positives (TP) were GO terms predicted above a certain reliability (RI) threshold (Method below), false positives (FP) were GO terms predicted but not annotated, and false negatives (FN) were GO terms annotated but not predicted. Based on these three numbers, we calculated precision (Eq. 1), recall (Eq. 2), and F1 score (Eq. 3) as follows.

$$P = precision = \frac{TP}{TP + FP} \tag{1}$$

$$R = recall = \frac{TP}{TP + FN} \tag{2}$$

$$F1 = 2 \bullet \frac{Precision \bullet Recall}{Precision + Recall} \tag{3}$$

The F_{max} value denoted the maximum F1 score achievable for any threshold in reliability (RI, Eq. 5). This implies that the assessment fixes the optimal value rather than the method providing this value. Although this arguably over-estimates performance, it has evolved to a quasi-standard of CAFA; the publicly available CAFA Assessment Tool[18,19] calculated F_{max} for the CAFA3 targets in the same manner as for the official CAFA3 evaluation. If not stated otherwise, we reported precision and recall values for the threshold leading to F_{max}.

CAFA3 assessed performance separately for two sets of proteins for all three ontologies: (i) proteins for which no experimental annotations were known beforehand (no-knowledge, NK evaluation mode) and (ii) proteins with some experimental annotations in one or two of the other ontologies (limited-knowledge, LK evaluation mode)[9,19]. We also considered these sets separately in our assessment. CAFA3 further distinguished between *full* and *partial evaluation* with *full evaluation* penalizing if no prediction was made for a certain protein, and *partial evaluation* restricting the assessment to the subset of proteins with predictions[19]. Our method predicted for every protein; thus, we considered only the *full evaluation*. Also following CAFA3, symmetric 95% confidence intervals were calculated as error estimates assuming a normal distribution and 10,000 bootstrap samples estimated mean and standard deviation.

Method: annotation transfer through embedding similarity. For a given query protein Q, GO terms were transferred from proteins with known GO terms (sets *GOA2020* and *GOA2017*) through an approach similar to the k-nearest neighbor algorithm (k-NN)[58]. For the query Q and for all proteins in, e.g., *GOA2020*, the SeqVec[20] embeddings were computed. Based on the Euclidean distance between two embeddings *n* and *m* (Eq. 4), we extracted the *k* closest hits to the query from the database where *k* constituted a free parameter to optimize.

$$d(n,m) = \sqrt{\sum_{i=1}^{1024} (n_i - m_i)^2} \tag{4}$$

In contrast to standard k-NN algorithms, all annotations from all hits were transferred to the query instead of only the most frequent one[58]. When multiple pairs reached the same distance, all were considered, i.e., for a given *k*, more than *k* proteins might be considered for the GO term prediction. The calculation of the pairwise Euclidean distances between queries and all database proteins and the subsequent nearest neighbor extraction was accomplished very efficiently. For instance, the nearest-neighbor search of 1000 query proteins against GOA20* with about 300,000 proteins took on average only about 0.005 s per query on a single i7-6700 CPU, i.e., less than two minutes for all human proteins.

Converting the Euclidean distance enabled to introduce a reliability index (RI) ranging from 0 (weak prediction) to 1 (confident prediction) for each predicted GO term *p* as follows:

$$RI(p) = \frac{1}{k}\sum_{i=1}^{l} \frac{0.5}{0.5 + d(q, n_i)} \tag{5}$$

with k as the overall number of hits/neighbors, l as the number of hits annotated with the GO term p and the distance $d(q, n_i)$ between query and hit being calculated according to Eq. (4).

Proteins represented by an embedding identical to the query protein (d = 0) led to RI = 1. Since the RI also takes into account, how many proteins l in a list of k hits are annotated with a certain term p (Eq. 5), predicted terms annotated to more proteins (larger l) have a higher RI than terms annotated to fewer proteins (smaller l). As this approach accounts for the agreement of the annotations between the k hits, it requires the RI to be normalized by the number of considered neighbors k, making it not directly comparable for predictions based on different values for k. On top, if different embeddings are used to identify close proteins, RI values are not directly comparable, because embeddings might be on different scales.

Instead of assessing embedding proximity through the Euclidean distance, the embedding field typically uses the cosine distance (Eq. 6):

$$d_{cosine}(n, m) = 1 - \frac{\sum_{i=1}^{1024} n_i m_i}{\sqrt{\sum_{i=1}^{1024} n_i^2} \bullet \sqrt{\sum_{i=1}^{1024} m_i^2}} \tag{6}$$

Our initial assessment suggested cosine and Euclidean distance to perform alike, and we chose to use the metric more familiar to structural biologists, namely the Euclidean distance throughout this analysis.

GO term similarity. We measured the similarity between two sets of GO annotations A and B through the Jaccard index (Eq. 7) where $|A \cap B|$ is the number of GO terms present in both sets and $|A \cup B|$ is the number of GO terms present in at least one of the sets (duplicates are only counted once):

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{7}$$

Correlation analysis. We analyzed the correlation between sequence identity and embedding similarity through the Spearman's rank correlation coefficient because our data was neither distributed normally, nor were the two measures for similarity measures linear. In contrast to, e.g. Pearson correlation, Spearman does not assume a normal distribution and detects monotonic instead of linear relations[59,60].

Availability. GO term predictions using embedding similarity for a certain protein sequence can be performed through our publicly available webserver: https://embed.protein.properties/. The source code along with all embeddings for *GOA2020* and *GOA2017*, and the CAFA3 targets are also available on GitHub: https://github.com/Rostlab/goPredSim (more details in the repository). In addition to reproducing the results, the source code also allows calculating embedding similarity using cosine distance.

Data availability

The source code and the embedding sets for target proteins and lookup databases are publicly available as a GitHub repository. GO term predictions for the SARS-CoV-2 proteins are provided as additional files and in the GitHub repository.

Received: 7 September 2020; Accepted: 24 December 2020
Published online: 13 January 2021

References

1. Krebs, H. A. & Johnson, W. A. Metabolism of ketonic acids in animal tissues. *Biochem J* **31**, 645–660. https://doi.org/10.1042/bj0310645 (1937).
2. The Gene Ontology Consortium. The Gene Ontology Resource: 20 years and still GOing strong. *Nucleic Acids Res* **47**, D330–D338. https://doi.org/10.1093/nar/gky1055 (2019).
3. UniProt Consortium. UniProt: a worldwide hub of protein knowledge. *Nucleic Acids Res.* **47**, D506–D515. https://doi.org/10.1093/nar/gky1049 (2019).
4. Hirst, J. D. & Sternberg, M. J. E. Prediction of structural and functional features of protein and nucleic acid sequences by artificial neural networks. *Biochemistry* **31**, 615–623 (1992).
5. Rost, B., Liu, J., Nair, R., Wrzeszczynski, K. O. & Ofran, Y. Automatic prediction of protein function. *Cell. Mol. Life Sci.* **60**, 2637–2650 (2003).
6. Leslie, C., Eskin, E., Weston, J. & Noble, W. S. Mismatch string kernels for SVM protein classification. *Bioinformatics*, in press (2003).
7. Ofran, Y., Punta, M., Schneider, R. & Rost, B. Beyond annotation transfer by homology: novel protein-function prediction methods to assist drug discovery. *Drug Discov. Today* **10**, 1475–1482 (2005).
8. Hamp, T. *et al.* Homology-based inference sets the bar high for protein function prediction. *BMC Bioinform.* **14**(Suppl 3), S 7. https://doi.org/10.1186/1471-2105-14-S3-S7 (2013).
9. Radivojac, P. *et al.* A large-scale evaluation of computational protein function prediction. *Nat. Methods* **10**, 221–227. https://doi.org/10.1038/nmeth.2340 (2013).
10. Cozzetto, D., Minneci, F., Currant, H. & Jones, D. T. FFPred 3: feature-based function prediction for all Gene Ontology domains. *Sci. Rep.* **6**, 31865. https://doi.org/10.1038/srep31865 (2016).
11. Kulmanov, M. & Hoehndorf, R. DeepGOPlus: improved protein function prediction from sequence. *Bioinformatics* **36**, 422–429. https://doi.org/10.1093/bioinformatics/btz595 (2020).
12. Zuckerkandl, E. Evolutionary processes and evolutionary noise at the molecular level. *J. Mol. Evol.* **7**, 269–311 (1976).
13. Nakai, K. & Horton, P. PSORT: a program for detecting sorting signals in proteins and predicting their subcellular localization. *Trends Biochem. Sci.* **24**, 34–36 (1999).

14. Nair, R. & Rost, B. Sequence conserved for sub-cellular localization. *Protein Sci.* **11**, 2836–2847 (2002).
15. Goldberg, T. *et al.* LocTree3 prediction of localization. *Nucleic Acids Res.* **42**, W350-355. https://doi.org/10.1093/nar/gku396 (2014).
16. Qiu, J. *et al.* ProNA2020 predicts protein-DNA, protein-RNA, and protein-protein binding proteins and residues from sequence. *J. Mol. Biol.* **432**, 2428–2443. https://doi.org/10.1016/j.jmb.2020.02.026 (2020).
17. Goldberg, T., Rost, B. & Bromberg, Y. Computational prediction shines light on type III secretion origins. *Sci. Rep.* **6**, 34516. https://doi.org/10.1038/srep34516 (2016).
18. Jiang, Y. *et al.* An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biol.* **17**, 184. https://doi.org/10.1186/s13059-016-1037-6 (2016).
19. Zhou, N. *et al.* The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biol.* **20**, 244. https://doi.org/10.1186/s13059-019-1835-8 (2019).
20. Heinzinger, M. *et al.* Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinform.* **20**, 723. https://doi.org/10.1186/s12859-019-3220-8 (2019).
21. Elnaggar, A. *et al.* ProtTrans: towards cracking the language of life's code through self-supervised deep learning and high performance computing. *bioRxiv* (2020).
22. Mikolov, T., Cheng, K., Corrado, G. & Dean, J. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, 2013, Workshop Track Proceedings.*
23. Allen, C. & Hospedales, T. Analogies Explained: Towards Understanding Word Embeddings. In *Proceedings of the 36th International Conference on Machine Learning.* 223–231 (PMLR).
24. Brokos, G.-I., Malakasiotis, P. & Androutsopoulos, I. Using Centroids of Word Embeddings and Word Mover's Distance for Biomedical Document Retrieval in Question Answering. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing.* 114–118 (Association for Computational Linguistics).
25. Kusner, M. J., Sun, Y., Kolkin, N. I. & Weinberger, K. Q. From Word Embeddings to Document Distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37.*
26. Rives, A. *et al.* Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, 622803. https://doi.org/10.1101/622803 (2020).
27. Vig, J. *et al.* BERTology meets Biology: Interpreting Attention in Protein Language Models. *arXiv* (2020).
28. R Core Team (R Foundation for Statistical Computing, 2017).
29. Rost, B. Twilight zone of protein sequence alignments. *Protein Eng.* **12**, 85–94 (1999).
30. Rost, B. Enzyme function less conserved than anticipated. *J. Mol. Biol.* **318**, 595–608 (2002).
31. Mika, S. & Rost, B. Protein–protein interactions more conserved within species than across species. *PLoS Comput. Biol.* **2**, e79. https://doi.org/10.1371/journal.pcbi.0020079 (2006).
32. Clark, W. T. & Radivojac, P. Analysis of protein function and its prediction from amino acid sequence. *Proteins* **79**, 2086–2096. https://doi.org/10.1002/prot.23029 (2011).
33. Rost, B. Protein structures sustain evolutionary drift. *Fold Des.* **2**, S19–S24 (1997).
34. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410. https://doi.org/10.1016/S0022-2836(05)80360-2 (1990).
35. Steinegger, M. & Söding, J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.* **35**, 1026–1028. https://doi.org/10.1038/nbt.3988 (2017).
36. Waterhouse, A. *et al.* SWISS-MODEL: homology modelling of protein structures and complexes. *Nucleic Acids Res.* **46**, W296–W303. https://doi.org/10.1093/nar/gky427 (2018).
37. El-Mabrouk, N. & Slonim, D. K. ISMB 2020 proceedings. *Bioinformatics* **36**, i1–i2. https://doi.org/10.1093/bioinformatics/btaa537 (2020).
38. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* 4171–4186 (Association for Computational Linguistics).
39. Steinegger, M., Mirdita, M. & Söding, J. Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods* **16**, 603–606. https://doi.org/10.1038/s41592-019-0437-4 (2019).
40. Steinegger, M. & Söding, J. Clustering huge protein sequence sets in linear time. *Nat. Commun.* **9**, 2542. https://doi.org/10.1038/s41467-018-04964-5 (2018).
41. Anderson, J. B. *et al.* Clonal evolution and genome stability in a 2500-year-old fungal individual. *Proc. Biol. Sci.* **285**, 20182233. https://doi.org/10.1098/rspb.2018.2233 (2018).
42. O'Donoghue, S. I. *et al.* SARS-CoV-2 structural coverage map reveals state changes that disrupt host immunity. *bioRxiv* (2020).
43. Wei, Q., Khan, I. K., Ding, Z., Yerneni, S. & Kihara, D. NaviGO: interactive tool for visualization and functional similarity and coherence analysis with gene ontology. *BMC Bioinform.* **18**, 177. https://doi.org/10.1186/s12859-017-1600-5 (2017).
44. Peters, M. E. *et al.* Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* 2227–2237 (Association for Computational Linguistics).
45. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Comput.* **9**, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 (1997).
46. Mousa, A. & Schuller, B. Contextual Bidirectional Long Short-Term Memory Recurrent Neural Network Language Models: a generative approach to sentiment analysis. In *Proceedings of the 15th Conference of the {E}uropean Chapter of the Association for Computational Linguistics: Volume 1, Long Papers.* 1023–1032 (Association for Computational Linguistics).
47. Peters, M., Ammar, W., Bhagavatula, C. & Power, R. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 1756–1765 (Association for Computational Linguistics).
48. Kim, Y., Jernite, Y., Sontag, D. & Rush, A. M. Character-aware Neural Language Models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence.* (AAAI Press).
49. Shen, D. *et al.* Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 440–450 (Association for Computational Linguistics).
50. Conneau, A., Douwe, K., Schwenk, H., Barrault, L. & Bordes, A. Supervised Learning of Universal Sentence Representations From Natural Language Inference Data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.* 670–680 (Association for Computational Linguistics).
51. Vaswani, A. *et al.* Attention is All You Need. In *Neural information processing systems conference.* (eds I Guyon *et al.*) 5998–6008 (Curran Associates, Inc.).
52. Bahdanau, D., Cho, K. H. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate in *arXiv.*
53. Camon, E. *et al.* The Gene Ontology Annotation (GOA) database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Res.* **32**, D262–266. https://doi.org/10.1093/nar/gkh021 (2004).
54. Huntley, R. P. *et al.* The GOA database: gene Ontology annotation updates for 2015. *Nucleic Acids Res.* **43**, D1057–1063. https://doi.org/10.1093/nar/gku1113 (2015).
55. *GOA*, http://www.ebi.ac.uk/GOA (2020).

56. Boutet, E., Lieberherr, D., Tognolli, M., Schneider, M. & Bairoch, A. UniProtKB/Swiss-Prot. *Methods Mol. Biol.* **406**, 89–112. https://doi.org/10.1007/978-1-59745-535-0_4 (2007).
57. Fu, L., Niu, B., Zhu, Z., Wu, S. & Li, W. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics* **28**, 3150–3152. https://doi.org/10.1093/bioinformatics/bts565 (2012).
58. Cover, T. & Hart, P. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory* **13**, 21–27 (1967).
59. Dodge, Y. *The Concise Encyclopedia of Statistics 502–505* (Springer, New York, 2008).
60. Spearman, C. The Proof and Measurement of Association Between Two Things. *Am. J. Psychol.* **15**, 72–101 (1904).

Acknowledgements

Thanks to Tim Karl and Inga Weise (both TUM) for invaluable help with technical and administrative aspects of this work. Thanks to the organizers and participants of the CAFA challenge for their crucial contributions, in particular to Predrag Radivojac (Northeastern Boston), Iddo Friedberg (Iowa State Ames), Sean Mooney (U of Washington Seattle), Casey Green (U Penn Philadelphia, USA), Mark Wass (Kent, England), and Kimberly Reynolds (U Texas Dallas). Particular thanks to the maintainers of the Gene Ontology (in particular to Michael Ashburner—Cambridge and Suzanna Lewis – Berkeley), as well as, the Gene Ontology Annotation database (maintained by the team of Claire O'Donovan at the EBI) for their standardized vocabulary and curated data sets. Last, but not least, thanks to all maintainers of public databases and to all experimentalists who enabled this analysis by making their data publicly available. This work was supported by the Bavarian Ministry of Education through funding to the TUM, by a grant from the Alexander von Humboldt foundation through the German Ministry for Research and Education (BMBF: Bundesministerium für Bildung und Forschung), by a grant from BMBF (Software Campus, 01IS17049) as well as by a grant from Deutsche Forschungsgemeinschaft (DFG–GZ: RO1320/4–1). We gratefully acknowledge the support of NVIDIA Corporation with the donation of one Titan GPU used for this research.

Author contributions

M.L. implemented the method goPredSim and performed evaluations. M.H. provided SeqVec and ProtBert embeddings and contributed various ideas and comments to improve goPredSim and its assessment. M.L. and M.H. performed the major part of manuscript writing and figure generation. C.D. implemented the webserver allowing GO term predictions using our method through a web interface. T.O. had the initial idea to calculate correlation between sequence and embedding similarity and computed the combination of sequence- and embedding-based transfer. B.R. supervised and guided the work over the entire time and proofread the manuscript. All authors read and approved the final manuscript.

Funding

Open Access funding enabled and organized by Projekt DEAL.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-020-80786-0.

Correspondence and requests for materials should be addressed to M.L.

A.1.3. Marquet, Heinzinger *et al.*, Human Genetics (2021)

Human Genetics
https://doi.org/10.1007/s00439-021-02411-y

ORIGINAL INVESTIGATION

Embeddings from protein language models predict conservation and variant effects

Céline Marquet[1,2] · Michael Heinzinger[1,2] · Tobias Olenyi[1,2] · Christian Dallago[1,2] · Kyra Erckert[1,2] · Michael Bernhofer[1,2] · Dmitrii Nechaev[1,2] · Burkhard Rost[1,3,4]

Received: 1 June 2021 / Accepted: 6 December 2021

Abstract

The emergence of SARS-CoV-2 variants stressed the demand for tools allowing to interpret the effect of single amino acid variants (SAVs) on protein function. While Deep Mutational Scanning (DMS) sets continue to expand our understanding of the mutational landscape of single proteins, the results continue to challenge analyses. Protein Language Models (pLMs) use the latest deep learning (DL) algorithms to leverage growing databases of protein sequences. These methods learn to predict missing or masked amino acids from the context of entire sequence regions. Here, we used pLM representations (embeddings) to predict sequence conservation and SAV effects without multiple sequence alignments (MSAs). Embeddings alone predicted residue conservation almost as accurately from single sequences as ConSeq using MSAs (two-state Matthews Correlation Coefficient—MCC—for ProtT5 embeddings of 0.596 ± 0.006 vs. 0.608 ± 0.006 for ConSeq). Inputting the conservation prediction along with BLOSUM62 substitution scores and pLM mask reconstruction probabilities into a simplistic logistic regression (LR) ensemble for Variant Effect Score Prediction without Alignments (VESPA) predicted SAV effect magnitude without any optimization on DMS data. Comparing predictions for a standard set of 39 DMS experiments to other methods (incl. ESM-1v, DeepSequence, and GEMME) revealed our approach as competitive with the state-of-the-art (SOTA) methods using MSA input. No method outperformed all others, neither consistently nor statistically significantly, independently of the performance measure applied (Spearman and Pearson correlation). Finally, we investigated binary effect predictions on DMS experiments for four human proteins. Overall, embedding-based methods have become competitive with methods relying on MSAs for SAV effect prediction at a fraction of the costs in computing/energy. Our method predicted SAV effects for the entire human proteome (~ 20 k proteins) within 40 min on one Nvidia Quadro RTX 8000. All methods and data sets are freely available for local and online execution through bioembeddings.com, https://github.com/Rostlab/VESPA, and PredictProtein.

Abbreviations

AI	Artificial intelligence
AUC	Area under the curve
BFD	Big Fantastic Database
CNN	Convolutional neural network
DL	Deep learning
EI	Evolutionary information
DMS	Deep mutational scanning
FNN	Feed forward neural network
GoF	Gain-of-function SAV
LoF	Loss-of-function SAV
LM	Language model
LR	Logistic regression
MAVE	Multiplexed Assays of Variant Effect
MCC	Matthews correlation coefficient
ML	Machine learning
MSA	Multiple sequence alignments

Céline Marquet and Michael Heinzinger have contributed equally to this work.

✉ Céline Marquet
celine.marquet@tum.de
http://www.rostlab.org/

[1] Department of Informatics, Bioinformatics and Computational Biology - i12, TUM-Technical University of Munich, Boltzmannstr. 3, Garching, 85748 Munich, Germany

[2] TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany

[3] Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, Garching, 85748 Munich, Germany

[4] TUM School of Life Sciences Weihenstephan (TUM-WZW), Alte Akademie 8, Freising, Germany

Published online: 30 December 2021

NLP	Natural language processing
OMIM	Online Mendelian Inheritance in Man
PDB	Protein Data Bank
pLM	Protein Language Model (used here: ESM-1b/1v: ProtBERT: ProtT5)
PMD	Protein mutant database
ProtT5beff	Rule-based method developed here using ProtT5 embeddings to predict binary SAV effects from single sequences
ProtT5cons	Method developed here using ProtT5 embeddings to predict residue conservation from single sequences optimizing a CNN on the unchanged pre-trained ProtT5
ReLU	Rectified linear unit
ROC	Receiver-operating characteristic
SAV	Single amino acid variant (also known as SAAV or nsSNP: or missense mutation/variant)
SOTA	State-of-the-art
SSD	Solid State Drive
SVM	Support Vector Machine
VESPA	Method developed here for Variant Effect Score Prediction without Alignments
VESPAl	Light VESPA: less accurate but faster

Introduction

Many different resources capture SAV effects. Mutations in the Spike (S) surface protein of SARS-CoV-2 have widened the attention to the complex issue of protein variant effects (Korber et al. 2020; Laha et al. 2020; Mercatelli and Giorgi 2020; O'Donoghue et al. 2020). The ability to distinguish between beneficial (= gain of function, GoF), deleterious (= loss of function, LoF) and neutral single amino acid variants (SAVs; also referred to as SAAV, missense mutations, or non-synonymous Single Nucleotide Variants: nsSNVs) continues to be a key challenge toward understanding how SAVs affect proteins (Adzhubei et al. 2010; Bromberg and Rost 2007, 2009; Ng and Henikoff 2003; Studer et al. 2013; Wang and Moult 2001). Recently, an unprecedented amount of in vitro data describing the quantitative effects of SAVs on protein function has been produced through Multiplexed Assays of Variant Effect (MAVEs), such as deep mutational scans (DMS) (Fowler and Fields 2014; Weile and Roth 2018). However, a comprehensive atlas of in vitro variant effects for the entire human proteome still remains out of reach (AVE Alliance Founding Members 2020). Yet, even for the existing experiments, intrinsic problems remain: (1) In vitro DMS data capture SAV effects upon molecular function much better than those upon biological processes, e.g., disease implications may be covered in databases such as the Online Mendelian Inheritance in Man (OMIM) (Amberger et al. 2019), but not in MaveDB (Esposito et al. 2019). (2) The vast majority of proteins have several structural domains (Liu and Rost 2003, 2004a, b); hence, most are likely to have several different molecular functions. However, each experimental assay tends to measure the impact upon only one of those functions. (3) In vivo protein function might be impacted in several ways not reproducible by in vitro assays.

Evolutionary information from MSAs is most important to predict SAV effects. Many in silico methods try to narrow the gap between known sequences and unknown SAV effects; these include (by earliest publication date): PolyPhen/PolyPhen2 (Adzhubei et al. 2010; Ramensky et al. 2002), SIFT (Ng and Henikoff 2003; Sim et al. 2012), I-Mutant (Capriotti et al. 2005), SNAP/SNAP2 (Bromberg and Rost 2007; Hecht et al. 2015), MutationTaster (Schwarz et al. 2010), Evolutionary Action (Katsonis and Lichtarge 2014), CADD (Kircher et al. 2014), PON-P2 (Niroula et al. 2015), INPS (Fariselli et al. 2015), Envision (Gray et al. 2018), DeepSequence (Riesselman et al. 2018), GEMME (Laine et al. 2019), ESM-1v (Meier et al. 2021), and methods predicting rheostat positions susceptible to gradual effects (Miller et al. 2017). Of these, only Envision and DeepSequence trained on DMS experiments. Most others trained on sparsely annotated data sets such as disease-causing SAVs from OMIM (Amberger et al. 2019), or from databases such as the protein mutant database (PMD) (Kawabata et al. 1999; Nishikawa et al. 1994). While only some methods use sophisticated algorithms from machine learning (ML; SVM, FNN) or even artificial intelligence (AI; CNN), almost all rely on evolutionary information derived from multiple sequence alignments (MSAs) to predict variant effects. The combination of evolutionary information (EI) and ML/AI has long been established as a backbone of computational biology (Rost 1996; Rost and Sander 1992, 1993), now even allowing AlphaFold2 to predict protein structure at unprecedented levels of accuracy (Jumper et al. 2021). Nevertheless, for almost no other task is EI as crucial as for SAV effect prediction (Bromberg and Rost 2007). Although different sources of input information matter, when MSAs are available, they trump all other features (Hecht et al. 2015). Even models building on the simplest EI, e.g., the BLOSUM62 matrix condensing bio-physical constraints into a 20 × 20 substitution matrix (Ng and Henikoff 2003) with no distinction between E481K (amino acid E at residue position 481 mutated to amino acid K) and E484K (part of SARS-CoV-2 Delta variant), or a simple conservation weight (Reeb et al. 2020) with no distinction of D484Q and D484K, almost reach the performance of much more complex and seemingly *advanced* methods.

Embeddings capture language of life written in proteins. Every year, algorithms improve natural language processing (NLP), in particular by feeding large text corpora into Deep Learning (DL)-based Language Models

(LMs). These advances have been transferred to protein sequences by learning to predict masked or missing amino acids using large databases of raw protein sequences as input (Alley et al. 2019; Bepler and Berger 2019a, 2021; Elnaggar et al. 2021; Heinzinger et al. 2019; Madani et al. 2020; Ofer et al. 2021; Rao et al. 2020; Rives et al. 2021). Processing the information learned by such protein LMs (pLMs), e.g., by constructing 1024-dimensional vectors of the last hidden layers, yields a representation of protein sequences referred to as embeddings [Fig. 1 in (Elnaggar et al. 2021)]. Embeddings have succeeded as exclusive input to predicting secondary structure and subcellular location at performance levels almost reaching (Alley et al. 2019; Heinzinger et al. 2019; Rives et al. 2021) or even exceeding (Elnaggar et al. 2021; Littmann et al. 2021c; Stärk et al. 2021) state-of-the-art (SOTA) methods using EI from MSAs as input. Embeddings even succeed in substituting sequence similarity for homology-based annotation transfer (Littmann et al. 2021a, b) and in predicting the effect of mutations on protein–protein interactions (Zhou et al. 2020). The power of such embeddings has been increasing with the advance of algorithms (Bepler and Berger 2021; Elnaggar et al. 2021; Rives et al. 2021). ESM-1v demonstrated pre-trained pLMs predicting SAV effect without any supervision at state-of-the-art level on DMS data using solely mask reconstruction probabilities (Meier et al. 2021). Naturally, there will be some limit to such improvements. However, the advances over the last months prove that this limit had not been reached by the end of 2020.

Here, we analyzed ways of using embeddings from pre-trained pLMs to predict the effect of SAVs upon protein function with a focus on molecular function, using experimental data from DMS (Esposito et al. 2019) and PMD (Kawabata et al. 1999). The embeddings from the pre-trained pLMs were not altered or optimized to suit the subsequent 2nd step of supervised training on data sets with more limited annotations. In particular, we assessed two separate supervised prediction tasks: conservation and SAV effects. First, we utilized pre-trained pLMs (ProtBert, ProtT5, ESM-1b) as static feature encoders (without fine-tuning the pLMs) to derive input embeddings for developing a method predicting the conservation that we could read off a family of aligned sequences (MSA) for each residue without actually generating the MSA. Second, we trained a Logistic Regression (LR) ensemble to predict SAV effect using (2a) the predictions of the best conservation predictor (ProtT5cons) together with (2b) substitution scores of BLOSUM62 and (2c) substitution probabilities of the pLM ProtT5. While substitution probabilities alone already correlated with DMS scores, we observed that adding conservation predictions together with BLOSUM62 increased performance. The resulting model for Variant Effect Score Prediction without Alignments (VESPA) was competitive with more complex solutions in terms of correlation with experimental DMS scores and computational and environmental costs. Additionally, for a small drop in prediction performance, we created a computationally more efficient method, dubbed VESPA-light (or short: VESPAl), by excluding substitution probabilities to allow proteome-wide analysis to complete after the coffee break on a single machine (40 min for human proteome on one Nvidia Quadro RTX 8000).

Methods

Data sets

In total, we used five different datasets. *ConSurf10k* was used to train and evaluate a model on residue conservation prediction. *Eff10k* was used to train SAV effect prediction. *PMD4k* and *DMS4* were used as test sets to assess the prediction of binary SAV effects. The prediction of continuous effect scores was evaluated on *DMS39*.

***ConSurf10k* assessed conservation.** The method predicting residue conservation used *ConSurf-DB* (Ben Chorin et al. 2020). This resource provided sequences and conservation for 89,673 proteins. For all, experimental high-resolution three-dimensional (3D) structures were available in the Protein Data Bank (PDB) (Berman et al. 2000). As standard-of-truth for the conservation prediction, we used the values from ConSurf-DB generated using HMMER (Mistry et al. 2013), CD-HIT (Fu et al. 2012), and MAFFT-LINSi (Katoh and Standley 2013) to align proteins in the PDB (Burley et al. 2019). For proteins from families with over 50 proteins in the resulting MSA, an evolutionary rate at each residue position is computed and used along with the MSA to reconstruct a phylogenetic tree. The ConSurf-DB conservation scores ranged from 1 (most variable) to 9 (most conserved). The PISCES server (Wang and Dunbrack 2003) was used to redundancy reduce the data set, such that no pair of proteins had more than 25% pairwise sequence identity. We removed proteins with resolutions > 2.5 Å, those shorter than 40 residues, and those longer than 10,000 residues. The resulting data set (ConSurf10k) with 10,507 proteins (or domains) was randomly partitioned into training (9392 sequences), cross-training/validation (555), and test (519) sets.

***Eff10k* assessed SAV effects.** This dataset was taken from the SNAP2 development set (Hecht et al. 2015). It contained 100,737 binary SAV-effect annotations (neutral: 39,700, effect: 61,037) from 9594 proteins. The set was used to train an ensemble method for SAV effect prediction. For this, we replicated the cross-validation (CV) splits used to develop SNAP2 by enforcing that clusters of sequence-similar proteins were put into the same CV split. More specifically, we clustered all sequence-similar proteins (PSI-BLAST *E* value < 1e-3) using single-linkage clustering, i.e., all connected nodes (proteins) were put into the same cluster. By placing all proteins within one cluster into the same CV split and rotating the splits, such that every split was used exactly once for testing, we ascertained that no pair of proteins between train and test shared significant sequence similarity (PIDE). More details on the dataset are given in SNAP2 (Hecht et al. 2015).

***PMD4k* assessed binary SAV effects.** From Eff10k, we extracted annotations that were originally adopted from PMD ("no change" as "neutral"; annotations with any level of increase or decrease in function as "effect"). This yielded 51,817 binary annotated SAVs (neutral: 13,638, effect: 38,179) in 4061 proteins. PMD4k was exclusively used for testing. While these annotations were part of Eff10k, all performance estimates for PMD4k were reported only for the PMD annotations in the testing subsets of the cross-validation splits. As every protein in Eff10k (and PMD4k) was used exactly once for testing, we could ascertain that there was no significant (prediction by homology-based inference possible) sequence-similarity between PMD4k and our training splits.

***DMS4* sampled large-scale DMS in vitro experiments annotating binary SAV effects.** This set contained binary classifications (effect/non-effect) for four human proteins (corresponding genes: BRAC1, PTEN, TPMT, PPARG) generated previously (Reeb 2020). These were selected as they were the first proteins with comprehensive DMS experiments including synonymous variants (needed to map from continuous effect scores to binary effect vs. neutral) resulting in 15,621 SAV annotations (Findlay et al. 2018; Majithia et al. 2016; Matreyek et al. 2018). SAVs with beneficial effect (= gain of function) were excluded, because they disagree between experiments (Reeb et al. 2020). The continuous effect scores of the four DMS experiments were mapped to binary values (effect/neutral) by considering the 95% interval around the mean of all experimental measurements as neutral, and the 5% tails of the distribution as "effect", as described in more detail elsewhere (Reeb et al. 2020). In total, the set had 11,788 neutral SAVs and 3516 deleterious effect SAVs. Additionally, we used two other thresholds: the 90% interval from mean (8926 neutral vs. 4545 effect) and the 99% interval from mean (13,506 neutral vs. 1,548 SAVs effect).

***DMS39* collected DMS experiments annotating continuous SAV effects.** This set was used to assess whether the methods introduced here, although trained only on binary effect data from Eff10k, had captured continuous effect scales as measured by DMS. The set was a subset of 43 DMS experiments assembled for the development of DeepSequence (Riesselman et al. 2018). From the original compilation, we excluded an experiment on tRNA as it is not a protein, on the toxin–antitoxin complex as it comprises multiple proteins and removed experiments for which only double variants existed. DMS39 contained 135,665 SAV scores, in total. The number of SAVs per experiment varied substantially between the 39 with an average of 3625 SAVs/experiment, a median of 1962, a minimum of 21, and a maximum of 12,729. However, to avoid any additional biases in the comparison to other methods, we avoided any further filtering step.

Input features

For the prediction of residue conservation, all newly developed methods exclusively trained on embeddings from pre-trained pLMs without fine-tuning those (no gradient was backpropagated to the pLM). The predictions of the best-performing method for conservation prediction were used in a second step together with substitution scores from BLOSUM62 and substitution probabilities from ProtT5 as input features to predict binary SAV effects.

Embeddings from pLMs: For conservation prediction, we used embeddings from the following pLMs: *ProtBert* (Elnaggar et al. 2021) based on the NLP (Natural Language Processing) algorithm BERT (Devlin et al. 2019) trained on Big Fantastic Database (BFD) with over 2.3 million protein sequences (Steinegger and Söding 2018), ESM-1b (Rives et al. 2021) that is conceptually similar to (Prot)BERT (both use a Transformer encoder) but trained on UniRef50 (The UniProt Consortium 2021) and *ProtT5-XL-U50* (Elnaggar et al. 2021) (for simplicity referred to as *ProtT5*) based on the NLP sequence-to-sequence model T5 (transformer encoder–decoder architecture) (Raffel et al. 2020) trained on BFD and fine-tuned on Uniref50. All embeddings were obtained from the bio_embeddings pipeline (Dallago et al. 2021). As described in ProtTrans, only the encoder side of ProtT5 was used and embeddings were extracted in half-precision (Elnaggar et al. 2021). The per-residue embeddings were extracted from the last hidden layer of the models with size 1024 × L (1280 for ESM-1b), where L is the length of the protein sequence and 1024 (or 1280 for ESM-1b) is the dimension of the hidden states/embedding space of ESM-1b, ProtBert, and ProtT5.

Context-dependent substitution probabilities: The training objective of most pLMs is to reconstruct corrupted amino acids from their non-corrupted protein sequence context. Repeating this task on billions of sequences allows pLMs to learn a probability of how likely it is to observe a token (an amino acid) at a certain position in the protein sequence. After pre-training, those probabilities can be extracted from pLMs by masking/corrupting one token/amino acid at a time, letting the model reconstruct it based on non-corrupted sequence context and repeating this for each token/amino acid in the sequence. For each protein, this gives a vector of length L by 20 with L being the protein's

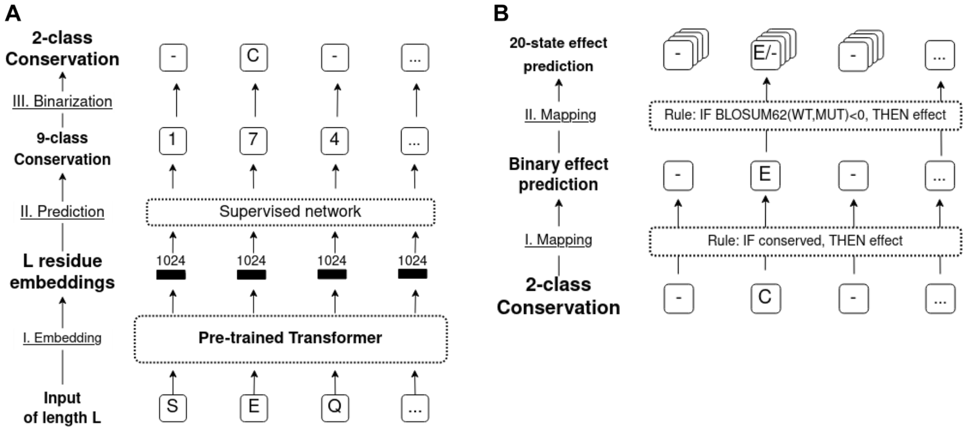

Fig. 1 Sketch of methods. Panel A sketches the conservation prediction pipeline: (I) embed protein sequence ("SEQ") using a pLM [here: ProtBERT, ProtT5 (Elnaggar et al. 2021) or ESM-1b (Meier et al. 2021)]. (II) Input embedding into supervised method (here: logistic regression, FNN or CNN) to predict conservation in 9-classes as defined by ConSurf-DB (Ben Chorin et al. 2020). (III) Map nine-class predictions > 5 to *conserved* (C), others to *non-conserved* (−). Panel B shows the use of binary conservation predictions as input to SAV effect prediction by (I) considering all residue positions predicted as conserved (C) as effect (E), all others as neutral (ProtT5cons-19equal and ConSeq-19equal). (II) Residues predicted as conserved are further split into specific substitutions (SAVs) predicted to have an effect (E) or not (−) if the corresponding BLOSUM62 score is < 0, all others are predicted as neutral (ProtT5-beff, ConSeq-BLOSUM62)

length and 20 being the probability distribution over the 20 standard amino acids. It was shown recently (Meier et al. 2021) that these probabilities provide a context-aware estimate for the effect of SAVs, i.e., the reconstruction probabilities depend on the protein sequence, and other methods have made use of similar probabilities (Hopf et al. 2017; Riesselman et al. 2018). To generate input features for our SAV effect predictor, we used, as suggested by Meier et al. (2021), the log-odds ratio between the probability of observing the wild-type amino acid at a certain position and the probability of observing a specific mutant at the same position: $\log\left(p\left(X_{i,mutant}\right)\right) - \log\left(p\left(X_{i,wildtype}\right)\right)$. The term $p\left(X_{i,mutant}\right)$ described the probability of an SAV occurring at position i and $p\left(X_{i,wildtype}\right)$ described the corresponding probability of the wild-type occurrence (native amino acid). To extract these probabilities for SAV effect prediction, we only considered the pLM embeddings correlating best with conservation (ProtT5). Additionally, we extracted probabilities for ProtBert on ConSurf10k to analyze in more detail the mistakes that ProtBert makes during reconstruction (SOM Fig. S5, S6).

Context-independent BLOSUM62 substitution scores: The BLOSUM substitution matrix gives a log-odds ratio for observing an amino acid substitution irrespective of its position in the protein (Henikoff and Henikoff 1992), i.e., the substitution score will not depend on a specific protein or the position of a residue within a protein but rather focuses on bio-chemical and bio-physical properties of amino acids. Substitution scores in BLOSUM were derived from comparing the log-odds of amino acid substitutions among well-conserved protein families. Typically applied to align proteins, BLOSUM scores are also predictive of SAV effects (Ng and Henikoff 2003; Sruthi et al. 2020).

Method development

In our three-stage development, we first compared different combinations of network architectures and pLM embeddings to predict residue conservation. Next, we combined the best conservation prediction method with BLOSUM62 substitution scores to develop a simple rule-based prediction of binary SAV effects. In the third step, we combined the predicted conservation, BLOSUM62, and substitution probabilities to train a new method predicting SAV effects for binary data from Eff10k and applied this method to non-binary DMS data.

Conservation prediction (ProtT5cons, Fig. 1A): Using either ESM-1b, ProtBert, or ProtT5 embeddings as input (Fig. 1a), we trained three supervised classifiers to distinguish between nine *conservation classes* taken from ConSurf-DB (early stop when optimum reached for ConSurf10k validation set). The objective of this task was to learn the prediction of family conservation from ConSurf-DB (Ben Chorin et al. 2020) based on the nine conservation classes introduced by that method that range from 1 (variable) to 9 (conserved) for each residue in a protein, i.e., this task implied a multi-class per-residue prediction. Cross-entropy loss together with Adam (Kingma and Ba 2014) was used to optimize each network toward predicting one out of nine conservation classes for each residue in a protein (per-token/per-residue task).

The models were: (1) standard Logistic Regression (LR) with 9000 (9 k) free parameters; (2) feed-forward neural network (FNN; with two FNN layers connected through the so-called ReLU (rectified linear unit) activations (Fukushima 1969); dropout rate 0.25; 33 k free parameters); (3) standard convolutional neural network (CNN; with two convolutional layers with a window size of 7, connected through ReLU activations; dropout rate of 0.25; 231 k free parameters). To put the number of free parameters into perspective: the ConSurf10k data set contained about 2.7 million samples, i.e., an order of magnitude more samples than free parameters of the largest model. On top of the 9-class prediction, we implemented a binary classifier (*conserved/non-conserved*; threshold for projecting nine to two classes optimized on validation set). The best-performing model (CNN trained on ProtT5) was referred to as ProtT5cons.

Rule-based binary SAV effect prediction (ProtT5beff, Fig. 1B): For rule-based binary SAV effect (*effect/neutral*) prediction, we considered multiple approaches. The first and simplest approach was to introduce a threshold to the output of ProtT5cons (no optimization on SAV data). Here, we marked all residues predicted to be conserved (conservation score > 5) as "effect"; all others as "neutral". This first level treated all 19 non-native SAVs at one sequence position equally (referred to as "19equal" in tables and figures). To refine, we followed the lead of SIFT (Ng and Henikoff 2003) using the BLOSUM62 (Henikoff and Henikoff 1992) substitution scores. This led to the second rule-based method dubbed *BLOSUM62bin* which can be considered a naïve baseline: SAVs less likely than expected (negative values in BLOSUM62) were classified as "effect"; all others as "neutral". Next, we combined both rule-based classifiers to the third rule-based method, dubbed *ProtT5beff* ("effect" if ProtT5cons predicts conserved, i.e., value > 5, and BLOSUM62 negative, otherwise "neutral", Fig. 1b). This method predicted binary classifications (effect/neutral) of SAVs without using any experimental data on SAV effects for optimization by merging position-aware information from ProtT5cons and variant-aware information from BLOSUM62.

Supervised prediction of SAV effect scores (*VESPA* and *VESPAl)*: For variant effect score prediction without alignments (VESPA), we trained a balanced logistic regression (LR) ensemble method as implemented in SciKit (Pedregosa et al. 2011) on the cross-validation splits of Eff10k. We rotated the ten splits of Eff10k, such that each data split was used exactly once for testing, while all remaining splits were used for training. This resulted in ten individual LRs trained on separate datasets. All of those were forced to share the same hyper-parameters. The hyper-parameters that differed from SciKit's defaults were: (1) *balanced weights*: class weights were inversely proportional to class frequency in input data; (2) *maximum number of iterations taken for the solvers to converge* was set to 600. The learning objective of each was to predict the probability of binary class membership (effect/neutral). By averaging their output, we combined the ten LRs to an ensemble method: $VESPA = ensemble\ of\ LRs = \frac{1}{10}\sum_{i=1}^{10} LR_i$. The output of VESPA is bound to [0,1] and by introducing a threshold can be readily interpreted as a probability for an SAV to be "neutral" (VESPA < 0.5) or to have "effect" (VESPA ≥ 0.5). As input for VESPA, we used 11 features to derive one score for each SAV; nine were the position-specific conservation probabilities predicted by ProtT5cons; one was the variant-specific substitution score from BLOSUM62, the other the variant- and position-specific log-odds ratio of ProtT5's substitution probabilities. To reduce the computational costs of VESPA, we introduced the "light" version VESPAl using only conservation probabilities and BLOSUM62 as input and thereby circumventing the computationally more costly extraction of the log-odds ratio. Both VESPA and VESPAl were only optimized on binary effect data from Eff10k and never encountered continuous effect scores from DMS experiments during any optimization.

Evaluation

Conservation prediction—ProtT5cons: To put the performance of ProtT5cons into perspective, we generated ConSeq (Berezin et al. 2004) estimates for conservation through PredictProtein (Bernhofer et al. 2021) using MMseqs2 (Steinegger and Söding 2018) and PSI-BLAST (Altschul et al. 1997) to generate MSAs. These were "estimates" as opposed to the standard-of-truth from ConSurf-DB, because, although they actually generated entire MSAs, the method for MSA generation was "just" MMseqs2 as opposed to HMMER (Mistry et al. 2013), and MAFFT-LINSi (Katoh and Standley 2013) for ConSurf-DB and the computation of weights from the MSA also required less computing resources. A random baseline resulted from randomly shuffling ConSurf-DB values.

Binary effect prediction—ProtT5beff: To analyze the performance of VESPA and VESPAl, we compared results to SNAP2 (Hecht et al. 2015) at the default binary threshold (score > − 0.05, default value suggested in original publication) on PMD4k and DMS4. Furthermore, we evaluated the rule-based binary SAV effect prediction ProtT5beff on the same datasets. To assess to which extent performance of ProtT5beff could be attributed to mistakes in ProtT5cons, we replaced residue conservation from ProtT5cons with conservation scores from ConSeq and applied the same two rule-based approaches as explained above (*ConSeq 19equal*: conserved predictions at one sequence position were considered "effect" for all 19 non-native SAVs and *ConSeq blosum62*: only negative BLOSUM62 scores at residues predicted as conserved were considered "effect"; all others considered "neutral" with both using the same threshold in conservation as for our method, i.e., conservation > 5 for effect) for PMD4k and DMS4. This failed for 122 proteins on PMD4k (3% of PMD4k), because the MSAs were deemed too small. We also compared ProtT5beff to the baseline based only on BLOSUM62 with the same thresholds as above (BLOSUM62bin). Furthermore, we compared to SNAP2 at default binary threshold of effect: SNAP2 score > − 0.05 (default value suggested in original publication). SNAP2 failed for four of the PMD4k proteins (0.1% of PMD4k). For the random baseline, we randomly shuffled ground truth values for each PMD4k and DMS4.

Continuous effect prediction—VESPA: We evaluated the performance of VESPA and VESPAl on DMS39 comparing to MSA-based DeepSequence (Riesselman et al. 2018) and GEMME (Laine et al. 2019), and the pLM-based ESM-1v (Meier et al. 2021). Furthermore, we evaluated log-odds ratios from ProtT5's substitution probabilities and BLOSUM62 substitution scores as a baseline. The DeepSequence predictions were copied from the supplement to the original publication (Riesselman et al. 2018), GEMME correlation coefficients were provided by the authors, and ESM-1v predictions were replicated using the online repository of ESM-1v. We used the publicly available ESM-1v scripts to retrieve "*masked-marginals*" for each of the five ESM-1v models and averaged over their outputs, because this strategy gave best performance according to the authors. If a protein was longer than 1022 (the maximum sequence length that ESM-1v can process), we split the sequence into non-overlapping chunks of length 1022. VESPA, VESPAl, and ESM-1v predictions did not use MSAs and therefore provided results for the entire input sequences, while DeepSequence and GEMME were limited to residues to which enough other protein residues were aligned in the MSAs.

Performance measures: We applied the following standard performance measures:

$$\mathrm{Q2} = 100 \cdot \frac{\text{(Number of residues predicted correctly in 2 states)}}{\text{(Number of all residues)}}. \tag{1}$$

Q2 scores (Eq. 1) described both binary predictions (conservation and SAV effect). The same held for F1-scores (Eq. 6, 7) and MCC (Matthews Correlation Coefficient, Eq. 8). We defined conserved/effect as the positive class and non-conserved/neutral as the negative class (indices "+" for positive, "–" for negative) and used the standard abbreviations of TP (true positives: number of residues predicted and observed as conserved/effect), TN (true negatives: predicted and observed as non-conserved/neutral), FP (false positives: predicted conserved/effect, observed non-conserved/neutral), and FN (false negatives: predicted non-conserved/neutral, observed conserved/effect)

$$Accuracy_+ = Precision_+ = Positive\ Predicted\ Value = \frac{TP}{TP+FP} \tag{2}$$

$$Accuracy\_ = Precision\_ = NegativePredictedValue = \frac{TN}{TN+FN} \tag{3}$$

$$Coverage_+ = Recall_+ = Sensitivity = \frac{TP}{TP+FN} \tag{4}$$

$$Coverage\_ = Recall\_ = Specificity = \frac{TN}{TN+FP} \tag{5}$$

$$F1_+ = 100 \bullet 2 \bullet \frac{Precision_+ \bullet Recall_+}{Precision_+ + Recall_+} \tag{6}$$

$$F1\_ = 100 \bullet 2 \bullet \frac{Precision\_ \bullet Recall\_}{Precision\_ + Recall\_} \tag{7}$$

$$MCC = \frac{TP \bullet TN - FP \bullet FN}{\sqrt{(TP+FP) \bullet (TP+FN) \bullet (TN+FP) \bullet (TN+FN)}} \tag{8}$$

$$Q9 = 100 \bullet \frac{Number\ of\ residues\ predicted\ correctly\ in\ 9\ states}{Number\ of\ all\ residues}. \tag{9}$$

Q9 is exclusively used to measure performance for the prediction of nine classes of conservation taken from ConSurf-DB. Furthermore, we considered the Pearson correlation coefficient

$$r_P = \rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}, \tag{10}$$

and the Spearman correlation coefficient where raw scores (X, Y of Eq. 10) are converted to ranks

$$r_S = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{Xrg_X} \sigma_{rg_Y}} \tag{11}$$

for continuous effect prediction.

Error estimates: We estimated symmetric 95% confidence intervals (CI Eq. 12) for all metrics using bootstrapping (Efron et al. 1996) by computing 1.96* standard deviation (SD) of randomly selected variants from all test sets with replacement over $n = 1000$ bootstrap sets

$$CI = 1.96 \bullet SD = 1.96 \bullet \sqrt{\frac{\sum (y_i - \bar{y})^2}{n}}, \tag{12}$$

with y_i being the metric for each bootstrap sample and $\overline{y}$ the mean over all bootstrap samples. We considered differences in performance significant if two CIs did not overlap.

Probability entropy: To investigate the correlation between embeddings and conservation classes of ConSurf-DB, we computed the entropy of pLM substitution probabilities (p) as

$$Entropy(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i. \tag{13}$$

Results

We first showed that probabilities derived from pLMs sufficed for the prediction of residue conservation from pLM embeddings without using MSAs (data set *ConSurf10k*; method *ProtT5cons*). Next, we presented a non-parametric rule-based SAV effect prediction based on predicted conservation (IF "predicted conserved" THEN "predict effect"; method *ProtT5beff*). We refined the rule-based system through logistic regression (LR) to predict SAV effect on variants labeled with "effect" or "neutral" (data set *Eff10k*; methods *VESPA, VESPAl*). Finally, we established that these new methods trained on binary data (effect/neutral) from *Eff10k* correlated with continuous DMS experiments.

Embeddings predicted conservation: First, we established that protein Language Models (pLMs) capture information correlated with residue conservation without ever seeing any such labels. As a standard-of-truth, we extracted the categorical conservation scores ranging from 1 to 9 (9: highly conserved, 1: highly variable) from ConSurf-DB (Ben Chorin et al. 2020) for a non-redundant subset of proteins with experimentally known structures (data set ConSurf10k). Those conservation scores correlated with the mask reconstruction probabilities output by ProtBert (Fig. 2). More specifically, one amino acid was corrupted at a time and ProtBert reconstructed it from non-corrupted sequence context. For instance, when corrupting and reconstructing all residues in ConSurf10k (one residue at a time), ProtBert assigned a probability to the native and to each of the 19 non-native (SAVs) amino acids for each position in the protein. Using those "*substitution probabilities*", ProtBert correctly predicted the native amino acid in 45.3% of all cases compared to 9.4% for a random prediction of the most frequent amino acid (Fig. S4). The entropy of these probability distributions correlated slightly with conservation (Fig. 2, Spearman's R = -—0.374) although never trained on such labels.

Next, we established that residue conservation can be predicted directly from embeddings by training a supervised network on data from ConSurf-DB. We exclusively used embeddings of pre-trained pLMs (ProtT5, ProtBert (Elnaggar et al. 2021), ESM-1b (Rives et al. 2021)), as input to relatively simple machine learning models (Fig. 1). Even the simplistic logistic regression (LR) reached levels of performance within about 20% of ConSeq (Berezin et al. 2004) conservation scores, which were derived from MSAs generated by the fast alignment method MMseqs2 (Steinegger and Söding 2017) (Fig. 3). The top prediction used ProtT5 embeddings which consistently outperformed predictions from ESM-1b and ProtBERT embeddings. For all three types of embeddings, the CNN outperformed the FNN, and these outperformed the LR. Differences between ProtBert and ProtT5 were statistically significant (at the 95% confidence interval, Eq. 12), while improvements from ProtT5 over ESM-1b were mostly insignificant. The ranking of the embeddings and models remained stable across several performance measures ($F1_{effect}$, $F1_{neutral}$, MCC, Pearson correlation coefficient, Table S1).

ConSurf-DB (Ben Chorin et al. 2020) simplifies family conservation to a single digit integer (9: highly conserved, 1: highly variable). We further reduced these classes to a binary classification (conserved/non-conserved) to later

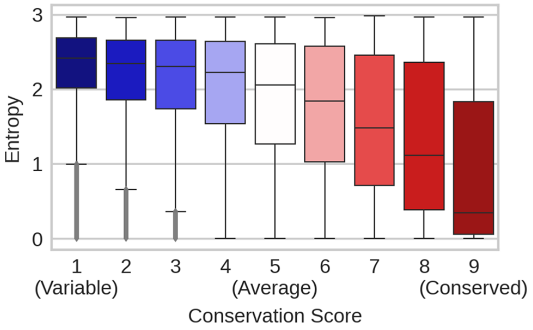

Fig. 2 pLMs captured conservation without supervised training or MSAs. ProtBert was optimized to reconstruct corrupted input tokens from non-corrupted sequence context (masked language modeling). Here, we corrupted and reconstructed all proteins in the ConSurf10k dataset, one residue at a time. For each residue position, ProtBert returned the probability for observing each of the 20 amino acids at that position. The higher one probability (and the lower the corresponding entropy), the more certain the pLM predicts the corresponding amino acid at this position from non-corrupted sequence context. Within the displayed boxplots, medians are depicted as black horizontal bars; whiskers are drawn at the 1.5 interquartile range. The x-axis gives categorical conservation scores (1: highly variable, 9: highly conserved) computed by ConSurf-DB (Ben Chorin et al. 2020) from multiple sequence alignments (MSAs); the y-axis gives the probability entropy (Eq. 13) computed without MSAs. The two were inversely proportional with a Spearman's correlation of -0.374 (Eq. 11), i.e., the more certain ProtBert's prediction, the lower the entropy and the higher the conservation for a certain residue position. Apparently, ProtBert had extracted information correlated with residue conservation during pre-training without having ever seen MSAs or any labeled data

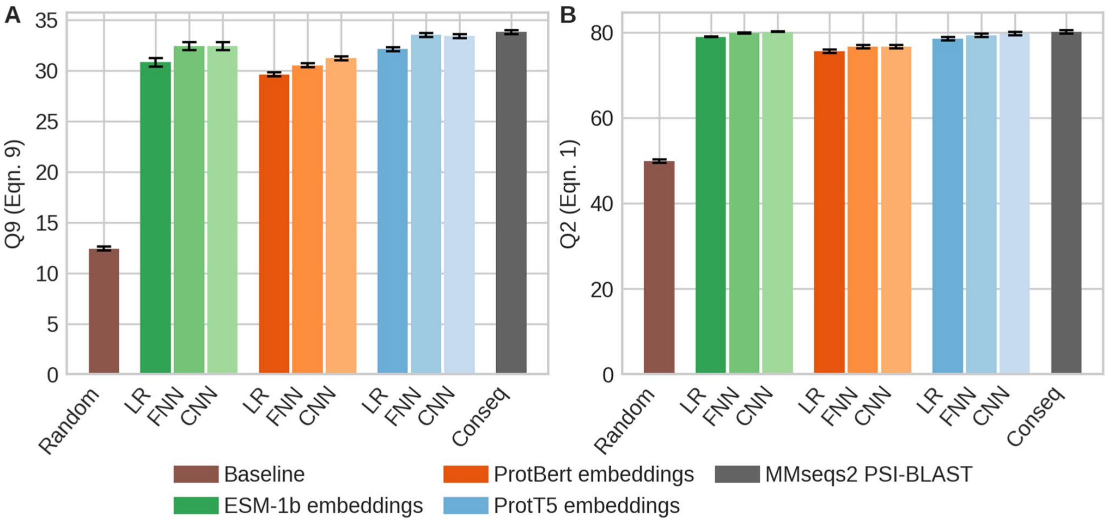

Fig. 3 Conservation predicted accurately from embeddings. Data: hold-out test set of *ConSurf10k* (519 sequences); panel A: nine-state per-residue accuracy (Q9, Eq. 9) in predicting conservation as defined by ConSurf-DB (Ben Chorin et al. 2020); panel B: two-state per-residue accuracy (Q2, Eq. 1; conservation score > 5: conserved, non-conserved otherwise). Supervised models (trained on *ConSurf10k*): **LR**: logistic regression (9,000 = 9 k free parameters), *FNN* feed-forward network (33 k parameters), and *CNN* convolutional neural network (231 k parameters with 0.25 dropout rate); methods: *ConSeq* computation of conservation weight through multiple sequence alignments (MSAs) (Berezin et al. 2004); *Random* random label swap. Model inputs were differentiated by color (green: ESM-1b embeddings (Rives et al. 2021), red: ProtBert embeddings (Elnaggar et al. 2021), blue: ProtT5 embeddings (Elnaggar et al. 2021), gray: MSAs (MMseqs2 (Steinegger and Söding 2017), and PSI-BLAST (Altschul et al. 1997)). Black whiskers mark the 95% confidence interval (± 1.96 SD; Eq. 12). ESM-1b and ProtT5 embeddings outperformed those from ProtBERT (Elnaggar et al. 2021); differences between ESM-1b and ProtT5 were not statistically significant, but ProtT5 consistently outperformed ESM-1b in all metrics but Q2 (Table S1). ESM-1b and ProtT5 as input to the CNN came closest to ConSeq (Table S1)

transfer information from conservation to binary SAV effect (effect/neutral) more readily. The optimal threshold for a binary conservation prediction was 5 (> 5 conserved, Fig. S1). However, performance was stable across a wide range of choices: between values from 4 to 7, MCC (Eq. 8) changed between 0.60 and 0.58, i.e., performance varied by 3.3% for 44.4% of all possible thresholds (Fig. S1). This was explained by the nine- and two-class confusion matrices (Fig. S2 and S3) for *ProtT5cons*, which showed that most mistakes were made between neighboring classes of similar conservation, or between the least conserved classes 1 and 2.

Conservation-based prediction of binary SAV effect better for DMS4 than for PMD4k? Next, we established that we could use the predicted conservation of ProtT5cons for rule-based binary SAV effect prediction without any further optimization and without any MSA. In using predicted conservation to proxy SAV effect, we chose the method best in conservation prediction, namely the CNN using ProtT5 embeddings (method dubbed *ProtT5cons*, Fig. 1B). The over-simplistic approach of considering any residue predicted as conserved to have an effect irrespective of the SAV (meaning: treat all 19 non-native SAVs alike) was referred to as “19equal”. We refined this rule-based approach by combining conservation prediction with a binary BLOSUM62 score (effect: if ProtT5cons predicted conserved AND BLOSUM62 < 0, neutral otherwise), which we referred to as *ProtT5beff*. For PMD4k, the following results were common to all measures reflecting aspects of precision and recall through a single number ($F1_{effect}$, $F1_{neutral}$ and MCC). First, the expert method SNAP2 trained on Eff10k (superset of PMD4k) achieved numerically higher values than all rule-based methods introduced here. Second, using the same SAV effect prediction for all 19 non-native SAVs consistently reached higher values than using the BLOSUM62 values (Fig. 4 and Table 1: *19equal* higher than *blosum62*). For some measures (Q2, $F1_{effect}$), values obtained using ConSeq for conservation (i.e., a method using MSAs) were higher than those for the ProtT5cons prediction (without using MSAs), while for others (MCC, $F1_{neutral}$**)**, this was reversed (Fig. 4, Table 1, Table S2).

Most performances differed substantially between PMD4k and DMS4, i.e., the first four proteins (BRAC1, PTEN, TPMT, and PPARG) for which we had obtained large-scale experimental DMS measures that could be converted into a binary scale (effect/neutral). First, using BLOSUM62 to convert ProtT5cons into SAV-specific predictions

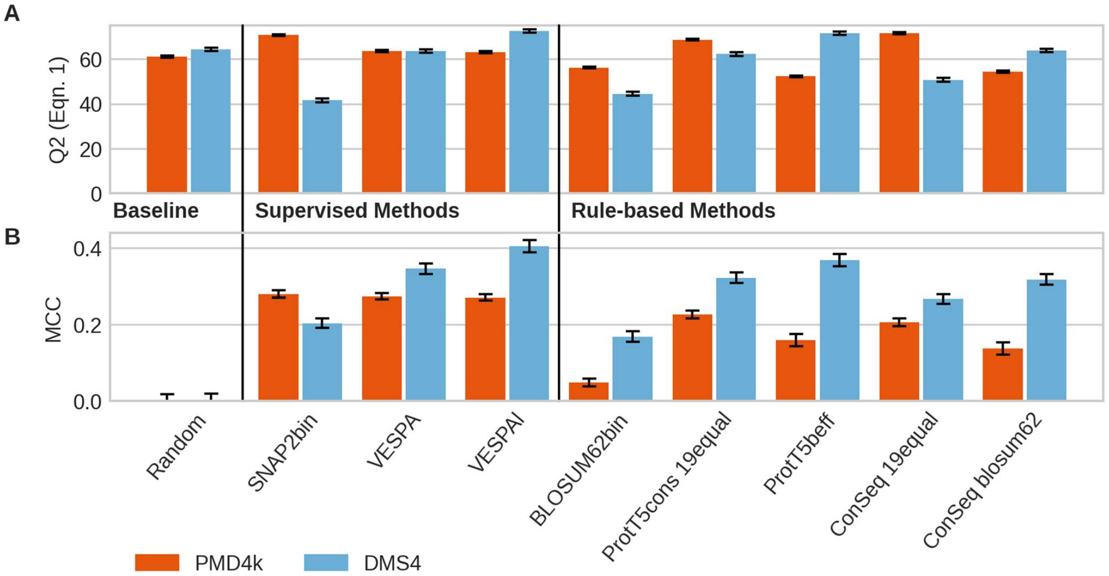

Fig. 4 Embedding-based binary SAV effect prediction is seemingly competitive. Data: *PMD4k* (red bars; 4 k proteins from PMD (Kawabata et al. 1999)); *DMS4* (blue bars) first four human proteins (BRAC1, PTEN, TPMT, PPARG) with comprehensive experimental DMS measurements including synonyms (here 95% threshold) (Reeb et al. 2020). Methods: SUPERVISED: **a** *SNAP2bin*: effect SNAP2 score > − 0.05, otherwise neutral; **b** *VESPA*: effect VESPA score > =0.5, otherwise neutral; **c** *VESPAl*: effect VESPAl score > =0.5, otherwise neutral. RULE-BASED: **d** *BLOSUM62bin*: irrespective of residue position, negative BLOSUM62 scores predicted as effect, others as neutral; **e** *ProtT5cons|ConSeq 19equal*: all 19 non-native SAVs predicted equally: effect if ProtT5cons|ConSeq predicted residue position to be conserved, otherwise neutral; **f** *ProtT5beff|ConSeq blosum62*: effect if ProtT5cons|ConSeq predicts conserved and BLOSUM62 negative, otherwise neutral. BASELINE: **g** *Random*: random shuffle of experimental labels. All values for DMS4 computed for binary (effect/neutral) mapping of experimental DMS values with panel A giving the two-state per-residue accuracy (Q2, Eq. 1) and panel B giving the Matthews Correlation Coefficient (MCC, Eq. 8). Error bars: Black bars mark the 95% confidence interval (±1.96 SD, Eq. 12). For all methods, the MCC differences between the two data sets PMD4k and DMS4 were statistically significant (exception: random)

outperformed the MSA-based conservation lookup from ConSeq, the expert method SNAP2 trained on PMD4k (Table 1: ProtT5beff highest rule-based), and the newly introduced VESPA. Second, combining the BLOSUM62 matrix with conservation also improved ConSeq (Table 1: ConSeq: *19equal* lower than *blosum62*). Third, ranking across different performance measures correlated much better than for PMD4k (Tables S1–S5). As the mapping from continuous DMS effect scores to binary labels might introduce systematic noise, we also investigated different thresholds for this mapping. However, results for DMS4 at intervals of 90% (Table S3) and 99% (Table S5) around the mean showed similar trends.

We trained a logistic regression (LR) ensemble (VESPA) on cross-validation splits replicated from the SNAP2 development set. For binary effect prediction, we introduced a threshold (≥ 0.5 effect, otherwise neutral) to the output scores of VESPA. When comparing VESPA and VESPAl (light version of VESPA) to the other methods on PMD4k, we observed a different picture than for the rule-based approaches. While SNAP2 still resulted in the highest MCC (0.28 ± 0.01), it was not significantly higher than that of VESPA and VESPAl (MCC: 0.274 ± 0.09 and 0.271 ± 0.09, respectively), and its development set overlapped with PMD4k. When evaluating the methods on DMS4, the best-performing method, VESPAl (MCC 0.405 ± 0.016), outperformed SNAP2 (MCC 0.204 ± 0.012) and VESPA (MCC 0.346 ± 0.014) as well as all rule-based methods (Table 1). We observed the same trends for other intervals (Tables S3–S5).

pLMs predicted SAV effect scores without MSAs. Could VESPA, trained on binary effect data (Eff10k) capture continuous SAV effect scores measured by DMS? For ease of comparison with other methods, we chose all 39 DMS experiments (DMS39) with single SAV effect data assembled for the development of DeepSequence (Riesselman et al. 2018). Several methods have recently been optimized on DMS data, e.g., the apparent state-of-art (SOTA), DeepSequence trained on the MSAs of each of those 39 experiments. Another recent method using evolutionary

Table 1 Performance in binary SAV effect prediction[a]

Data set	PMD4k		DMS4	
Method/metric	Q2 (Eq. 1)	MCC (Eq. 8)	Q2 (Eq. 1)	MCC (Eq. 8)
Random	61.08% ± 0.41	− 0.002 ± 0.016	64.27% ± 0.76	− 0.001 ± 0.018
Supervised methods				
SNAP2bin	70.66% ± 0.39	0.280 ± 0.010	41.55% ± 0.82	0.204 ± 0.012
VESPA	63.52% ± 0.43	0.274 ± 0.086	63.56% ± 0.79	0.346 ± 0.014
VESPAl	63.04% ± 0.43	0.271 ± 0.085	72.59% ± 0.72	**0.405 ± 0.016**
Rule-based methods				
BLOSUM62bin	56.17% ± 0.43	0.049 ± 0.010	44.47% ± 0.84	0.169 ± 0.014
ProtT5cons-19equal	68.58% ± 0.41	0.227 ± 0.010	62.20% ± 0.82	0.322 ± 0.014
ProtT5-beff	52.26% ± 0.43	0.160 ± 0.016	71.47% ± 0.75	0.369 ± 0.016
ConSeq-19equal	**71.51% ± 0.39**	0.206 ± 0.010	50.70% ± 0.84	0.267 ± 0.012
ConSeq blosum62	54.32% ± 0.43	0.138 ± 0.016	63.81% ± 0.8	0.318 ± 0.014

[a]Data sets: The *PMD4k* data set contained 4 k proteins from the PMD (Kawabata et al. 1999); 74% of the SAVs were deemed effect in a binary classification. *DMS4* marks the first four human proteins (BRAC1, PTEN, TPMT, PPARG) for which we obtained comprehensive experimental DMS measurements along with a means of converting experimental scores into a binary version (effect/neutral) using synonyms. DMS4 results are shown for a threshold of 95%: the continuous effect scores were binarized by assigning the middle 95% of effect scores as neutral variants and SAVs resulting in effect scores outside this range as effect variants (Reeb et al. 2020). Methods: *SNAP2bin*: effect SNAP2 score > − 0.05, otherwise neutral; *VESPA*: effect score ≥ 0.5, otherwise neutral; *VESPAl*: effect score ≥ 0.5, otherwise neutral; *BLOSUM62*: negative BLOSUM62 scores predicted as effect, others as neutral; *ProtT5cons|ConSeq-19equal*: all 19 non-native SAVs predicted equally: effect if ProtT5cons|ConSeq predicted/labeled as conserved, otherwise neutral; *ProtT5beff|ConSeq-blosum62*: effect if ProtT5cons|ConSeq predicted/labeled as conserved and BLOSUM62 negative, otherwise neutral. ± values mark the 95% confidence interval (Eq. 12). For each column, if available, significantly best results are highlighted in bold

information in a more advanced way than standard profiles from MSAs appears to reach a similar top level without machine learning, namely GEMME (Laine et al. 2019), and so does a method based on probabilities from pLMs, namely ESM-1v, without using MSAs. Comparing all those to VESPA, we could not observe a single method outperforming all others on all DMS39 experiments (Fig. 5). The four methods compared (two using MSAs: DeepSequence and GEMME, two using probabilities from pLMs instead of MSAs: ESM-1v and VESPA) reached Spearman rank correlations above 0.4 for 36 DMS experiments. In fact, for the 11 highest correlating out of the 39 experiments, predictions were as accurate as typically the agreement between two different experimental studies of the same protein (Spearman 0.65 (Reeb et al. 2020)).

GEMME had a slightly higher mean and median Spearman correlation (Eq. 11) than DeepSequence, ESM-1v, VESPA, and all others tested (Fig. 6A, Table 2). When considering the symmetric 95% confidence intervals (Eq. 12), almost all those differences were statistically insignificant (Fig. 6B) except for only using BLOSUM62. In terms of mean Spearman correlation, VESPA was slightly higher than DeepSequence, which was slightly higher than ESM-1v (Fig. 6A), but again neither was significantly better. The median Spearman correlation was equal for ESM-1v and VESPA and insignificantly lower for DeepSequence. The fastest method, VESPAl, reached lower Spearman correlations than all other major methods (Fig. 6). Ranking and relative performance after correcting for statistical significance were identical for Spearman and Pearson correlation (Table S6).

For comparison, we also introduced two advances on a random baseline, namely the raw BLOSUM62 scores and the raw ProtT5 log-odds scores (Fig. 6; Fig. S7). BLOSUM62 was consistently and statistically significantly outperformed by all methods, while the ProtT5 log-odds averages were consistently lower, albeit not with statistical significance. As pLM-based methods were independent of MSAs, they predicted SAV scores for all residues contained in the DMS39 data sets, while, e.g., DeepSequence and GEMME could predict only for the subset of the residues covered by large enough MSAs. This was reflected by decreased coverage of methods relying on MSAs (DeepSequence and GEMME; Table S8). The Spearman correlation of ESM-1v, VESPA, and VESPAl for the SAVs in regions without MSAs was significantly lower than that in regions with MSAs available (Table S7).

SAV effect predictions blazingly fast: One important advantage of predicting SAV effects without using MSAs is the computational efficiency. For instance, to predict the

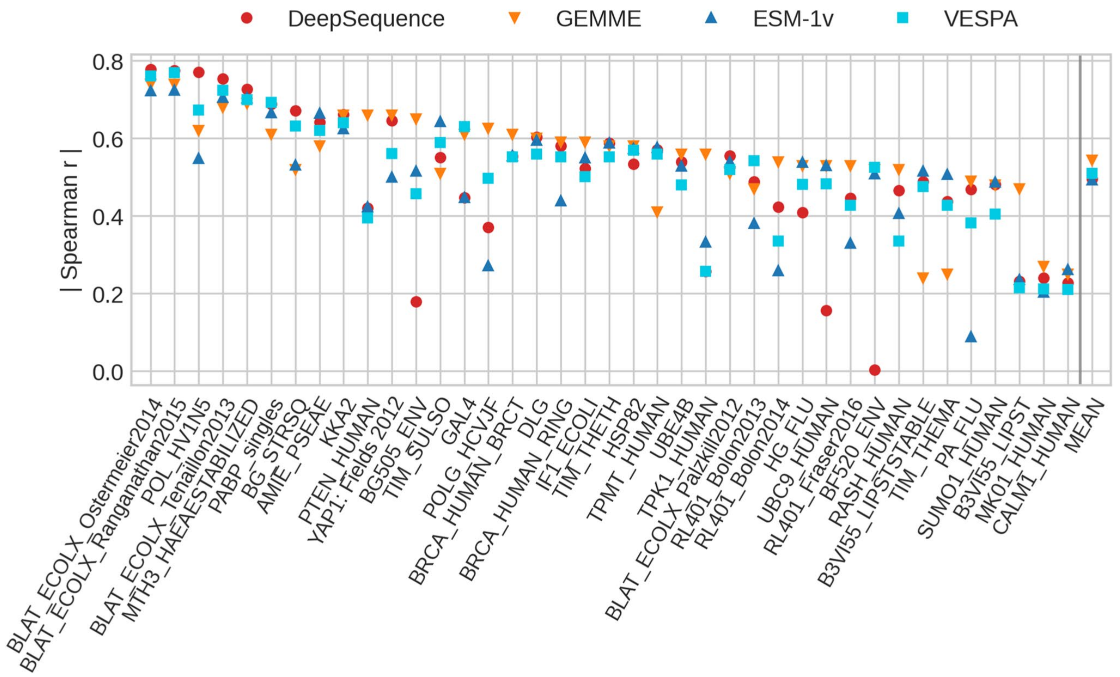

Fig. 5 No SAV effect prediction consistently best on DMS data. Data: *DMS39* (39 DMS experiments gathered for the development of DeepSequence (Riesselman et al. 2018)); experiments sorted by the maximum absolute Spearman coefficient for each experiment. Methods: **a** *DeepSequence* trained an unsupervised model for each DMS experiment using only MSA input, i.e., no effect score labels were used (Riesselman et al. 2018); **b** *GEMME* inferred evolutionary trees and conserved sites from MSAs to predict effects (Laine et al. 2019); **c** *ESM-1v* correlated log-odds of substitution probabilities (Methods) with SAV effect magnitudes (Meier et al. 2021); **d** *VESPA* (this work) trained a logistic regression ensemble on binary SAV classification (effect/neutral) using predicted conservation (ProtT5cons), BLOSUM62 (Henikoff and Henikoff 1992), and log-odds of substitution probabilities from ProtT5 (Elnaggar et al. 2021) as input (without any optimization on DMS data). The values for the absolute Spearman correlation (Eq. 11) are shown for each method and experiment. The rightmost column shows the mean absolute Spearman correlation for each method. Although some experiments correlated much better (toward left) with predictions than others (toward right), the spread between prediction methods appeared high for both extremes; DeepSequence was the only method reaching a correlation of 0 for one experiment; another one and three experiments were predicted with correlations below 0.2 for ESM-1v and DeepSequence, respectively, while the vast number of the 4×39 predictions reached correlations above 0.4

mutational effects for all 19 non-native SAVs in the entire human proteome (all residues in all human proteins) took 40 min on one Nvidia Quadro RTX 8000 using VESPAl. In turn, this was 40 min more than using BLOSUM62 alone (nearly instantaneous), but this instantaneous BLOSUM62-based prediction was also much worse (Q2 for binary BLOSUM62 prediction worse than random, Table 1). In contrast, running methods such as SNAP2 (or ConSeq) required first to generate MSAs. Even the blazingly fast MMseqs2 (Steinegger and Söding 2017) needed about 90 min using batch-processing on an Intel Skylake Gold 6248 processor with 40 threads, SSD and 377 GB main memory. While VESPAl computed prediction scores within minutes for an entire proteome, VESPA and ESM-1v require minutes for some single proteins depending on sequence length, e.g., ESM-1v took on average 170 s per protein for the DMS39 set, while ProtT5 required on average 780 s. This originated from the number of forward passes required to derive predictions: while VESPAl needed only a single forward pass through the pLM to derive embeddings for conservation prediction, VESPA and ESM-1v (when deriving "masked-marginals" as recommended by the authors) required L forward passes with L being the protein length, because they corrupt one amino acid at a time and try to reconstruct it. The large difference in runtime between ESM-1v and ProtT5 originated from the fact that ESM-1v cropped sequences after 1022, reducing the strong impact of outliers, i.e., runtime of transformer-based models scales quadratically with

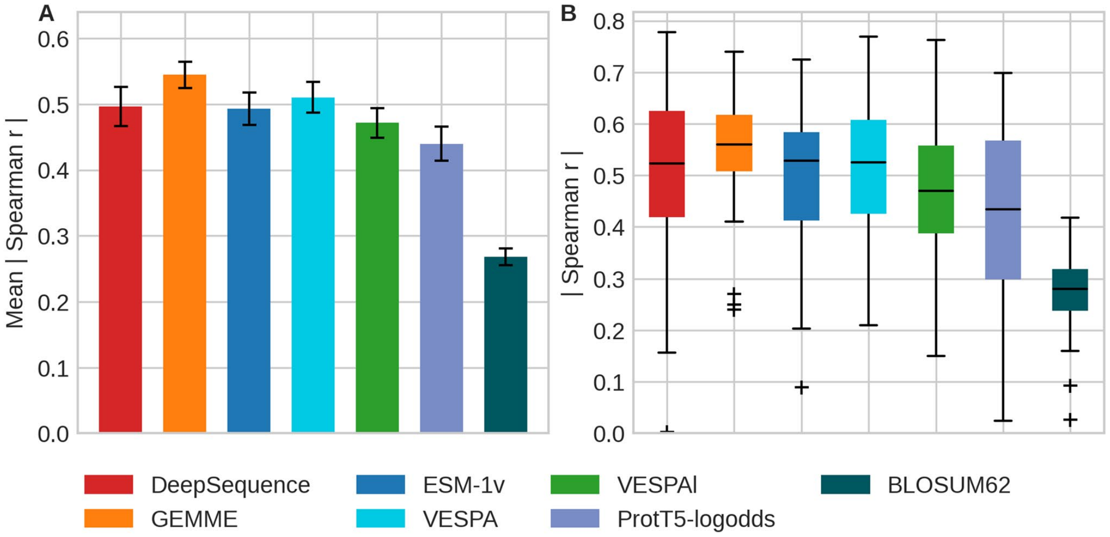

Fig. 6 Spearman correlation between prediction and DMS experiment varied. Data and methods as for Fig. 5 with addition of: *VESPAl*: fast version of VESPA with input limited to ProtT5cons and BLOSUM62; *ProtT5-logodds*: raw log-odds from ProtT5 embeddings (Elnaggar et al. 2021); and raw *BLOSUM62* substitution scores (Henikoff and Henikoff 1992). Panel A: mean absolute Spearman correlation coefficient (Eq. 11) for each method over all 39 DMS experiments; error bars highlight 0.95 confidence interval (1.96 standard errors). Ignoring statistical significance, the numerical ranking would be: GEMME, VESPA, DeepSequence, ESM-1v, VESPAl, ProtT5-logodds, and BLOSUM62. However, the first four did not differ by any statistical significance, and while those ranked 5 and 6 differed from the best four, 5 was close to 4, and 6 close to 5; only BLOSUM62, the raw substitution scores compiled as background were clearly worst. Panel B: boxplots on absolute Spearman correlation coefficients (Eq. 11) for each method over the 39 DMS experiments. The medians are depicted as black horizontal bars; whiskers are drawn at the 1.5 interquartile range

sequence length, so while the shortest protein (71 residues) in the DMS39 set took only 5 s to compute, the longest (3033 residues) took 4.5 h to compute. We leave investigating the effect of splitting very long proteins into (overlapping) chunks to future work.

Discussion

Conservation predicted by embeddings without MSAs. Even a simple logistic regression (LR) sufficed to predict per-residue conservation values from raw embeddings without using MSAs (Fig. 3, Table S1). Relatively shallow CNNs (with almost 100-times fewer free parameters than samples despite early stopping) improved over the LR to levels in predicting conservation only slightly below conservation assigned by ConSeq which explicitly uses MSAs (Fig. 3, Table S1). Did this imply that the pLMs extracted evolutionary information from unlabeled sequence databases (BFD (Steinegger and Söding 2018) and UniProt (The UniProt Consortium 2021))? The answer might be more elusive than it seems. The methodology (pLMs) applied to predict conservation never encountered any explicit information about protein families through MSAs, i.e., the pLMs used here never had an explicit opportunity to pick up evolutionary constraints from related proteins. The correlation between substitution probabilities derived from pLMs and conservation (Fig. 2) might suggest that pLMs implicitly learned evolutionary information.

A possible counterargument builds around the likelihood to pick up evolutionary constraints. The pLM clearly learned the reconstruction of more frequent amino acids much better than that of less frequent ones (Fig. S5). Unsurprisingly, AI is pushed most in the direction of most data. In fact, the differences between amino acid compositions were relatively small (less than factor of 10), suggesting that even an event occurring at one-tenth of the time may challenge pLMs. If the same pLM has to learn the evolutionary relation between two proteins belonging to the same family, it has to effectively master an event happening once in a million (assuming an average family size of about 2.5 k—thousand—in a database with 2.5b—billion—sequences). How can the model trip over a factor of 10^1 and at the same time master a factor of 10^6? Indeed, it seems almost impossible. If so, the pLM may not have learned evolutionary constraints, but the type of *bio-physical* constraint that also constrain evolution. In this interpretation, the pLM did not learn evolution, but

Table 2 Spearman correlation between SAV effect prediction and DMS experiments[a]

Method	Mean absolute r_S (Eq. 11)	Median absolute r_S (Eq. 11)
MSA-based		
DeepSequence	0.50 ± 0.03	0.52 ± 0.03
GEMME	0.53 ± 0.02	0.56 ± 0.02
pLM-based		
ESM-1v	0.49 ± 0.02	0.53 ± 0.02
VESPA	0.51 ± 0.02	0.53 ± 0.02
VESPAl	0.47 ± 0.02	0.47 ± 0.02

[a]Data sets: *DMS39* [39 DMS experiments gathered for the development of DeepSequence (Riesselman et al. 2018)] with 135,665 SAV scores. Methods: *DeepSequence*: AI trained on MSA for each of the DMS experiments (Riesselman et al. 2018); *GEMME*: using evolutionary information calculated from MSAs with few parameters optimized on DMS (Laine et al. 2019); *ESM-1v*: embedding-based prediction methods (Meier et al. 2021); *VESPA*: method developed here using logistic regression to combine predicted conservation (ProtT5cons), BLOSUM62 (Henikoff and Henikoff 1992) substitution scores, and log-odds from ProtT5 (Elnaggar et al. 2021); *VESPAl*: "light" version of VESPA using only predicted conservation and BLOSUM62 as input. ± values mark the standard error

the constraints "written into protein sequences" that determine which residue positions are more constrained.

In fact, one pLM used here, namely ProtT5, has recently been shown to explicitly capture aspects of long-range inter-residue distances directly during pre-training, i.e., without ever being trained on any labeled data pLMs pick up structural constraints that allow protein 3D structure prediction from single protein sequences (Weißenow et al. 2021). Another explanation for how ProtT5 embeddings capture conservation might be that pLMs picked up signals from short, frequently re-occurring sequence/structure motifs such as localization signals or catalytic sites that are more conserved than other parts of the sequence. If so, the pLM would not have to learn relationship between proteins but only between fragments, thereof reducing the factor 10^6 substantially. We could conceive of these motifs resembling some evolutionary nuclei, i.e., fragments shorter than structural domains that drove evolution (Alva et al. 2015; Ben-Tal and Lupas 2021; Kolodny 2021). Clearly, more work will have to shed light on the efficiency of (p)LMs in general (Bommasani et al. 2021).

Transformer-based pLMs best? We have tested a limited set of pLMs, largely chosen, because those had appeared to perform better than many other methods for a variety of different prediction tasks. Does the fact that in our hands Transformer-based pLMs worked best to predict residue conservation and SAVs imply that those will generally outperform other model types? By no means. While we expect that the about twenty approaches that we have compared in several of our recent methods (including the following 13: ESM-1[b|v] Meier et al. 2021; Rives et al. 2021), ProSE[*|DLM|MT] (Bepler and Berger 2019b, 2021), Prot[Albert|Bert|Electra|Vec|T5|T5XL|T5XLNet|T5XXL] (Elnaggar et al. 2021; Heinzinger et al. 2019) provided a somehow representative sampling of the existing options, our conclusions were only valid for embeddings extracted in a generic way from generic pLMs without any bearing on the methods underlying those pLMs.

Predicted conservation informative about SAV effects: DMS data sets with comprehensive experimental probing of the mutability landscape (Hecht et al. 2013) as, e.g., collected by MaveDB (Esposito et al. 2019) continue to pose problems for analysis, possibly due to a diversity of assays and protocols (Livesey and Marsh 2020; Reeb et al. 2020). Nevertheless, many such data sets capture important aspects about the susceptibility to change, i.e., the mutability landscape (Hecht et al. 2013). As always, the more carefully selected data sets become, the more they are used for the development of methods and therefore no longer can serve as independent data for assessments (Grimm et al. 2015; Reeb et al. 2016). Avoiding the traps of circularity and over-fitting by skipping training, our non-parametric rule-based approaches (ProtT5cons and ProtT5beff) suggested that predictions of SAV effects (by simply assigning "effect" to those SAVs where ProtT5cons predicted conserved and the corresponding BLOSUM62 value was negative) outperformed ConSeq with MSAs using the same idea, and even the expert effect prediction method SNAP2 (Fig. 4, Table 1).

Strictly speaking, it might be argued that one single free parameter was optimized using the data set, because for the PMD4k data set, the version that predicted the same effect for all 19-SAVs appeared to outperform the SAV-specific prediction using BLOSUM62 (*19equal* vs *blosum62* in Fig. 4 and Table 1). However, not even the values computed for PMD4k could distract from the simple fact that not all SAVs are equal, i.e., that regardless of model performance, *19equal* will not be used exclusively for any method. In fact, the concept of combining predictions with BLOSUM62 values has been shown to succeed for function prediction before (Bromberg and Rost 2008; Schelling et al. 2018) in that sense it was arguably not an optimizable hyperparameter. Embeddings predicted conservation (Fig. 3); conservation predicted SAV effects (Fig. 4). Did this imply that embeddings captured evolutionary information? Once again, we could not answer this question either way directly. To repeat: our procedure/method never used information from MSAs in any way. Could it have implicitly learned this? To repeat the previous speculation: embeddings *might* capture a reality that constrains what can be observed in evolution, and this reality is exactly what is used for the part of the SAV effect prediction that succeeds. If so, we would argue that our simplified method did not succeed, because it predicted conservation without using MSAs, but that it captured positions biophysically "marked by constraints", i.e., residues with higher contact density in protein 3D structures (Weißenow et al. 2021). This assumption would explain how predicted conservation (ProtT5cons) not using evolutionary information could predict SAV effects better than a slightly more correct approach (ConSeq) using MSAs to extract evolutionary information (Fig. 4: ProtT5cons vs. ConSeq).

Substitution probabilities from pLMs capture aspects measured by DMS experiments: Using embeddings to predict SAV effects through conservation prediction succeeded but appeared like a detour. ESM-1v (Meier et al. 2021) pioneered a direct path from reconstruction/substitution probabilities of pLMs to SAV effect predictions. When comparing the ESM-1v encoder-based with the ProtT5 encoder–decoder-based Transformer, we encountered surprising results. Previously, ProtT5 usually performed at least on par with previous versions of ESM (e.g., ESM-1b (Rives et al. 2021)) or outperformed them (Elnaggar et al. 2021). In contrast, the substitution probabilities of ProtT5 were clearly inferior to those from ESM-1v in their correlation with the 39 DMS experiments (Fig. 6). This reversed trend might have resulted from a combination of the following facts: (1) ProtT5 is a single model, while ESM-1v is an ensemble of five pLMs potentially leading to a smoother substitution score. (2) ESM-1v was trained on UniRef90 instead of BFD/UniRef50 (ProtT5) possibly providing a broader view on the mutability landscape of proteins. In fact, the ESM-1v authors showed a significant improvement when pre-training on UniRef90 instead of UniRef50 (Rives et al. 2021). (3) ESM-1v is a BERT-style, encoder-based Transformer, while ProtT5 is based on T5's encoder-decoder structure. In previous experiments (Elnaggar et al. 2021), we only extracted embeddings from ProtT5's encoder (e.g., ProtT5cons is based on encoder embeddings), because its decoder fell significantly short in all experiments. However, only T5's decoder can output probabilities, so we had to fall back to ProtT5's decoder for SAV effect predictions. This discrepancy of encoder and decoder performance can only be sketched here. In short, encoder-based transformer models always *see* the context of the whole sequence (as does ProtT5 's encoder and ESM-1v), while decoder-based transformer models (such as ProtT5's decoder or GPT (Radford et al. 2019)) *see* only single-sided context, because they are generating text (sequence-to-sequence models (Sutskever et al. 2014)). This is crucial for translation tasks, but appeared sub-optimal in our setting. Despite this shortcoming in performance, we trained VESPA based on log-odds derived from ProtT5 substitution probabilities, mainly because we started this work before the release of ESM-1v. Also, we hoped for synergy effects when implementing VESPA into the PredictProtein webserver, because ProtT5 is already used by many of our predictors. Finding the best combination of pLM substitution probabilities for SAV effect prediction will remain subject for future work.

Fast predictions save computing resources? Our simple protocol introduced here enabled extremely efficient, speedy predictions. While pre-training pLMs consumed immense resources (Elnaggar et al. 2021), this was done in the past. The new development here was the models for the 2nd level supervised transfer learning. Inputting ProtT5 embeddings to predict residue conservation (ProtT5cons) or SAV effects (VESPA/VESPAl) for predictions in the future will consume very little additional resources. When running prediction servers such as PredictProtein (Bernhofer et al. 2021) queried over 3000 times every month, such investments could be recovered rapidly at seemingly small prices to pay even if performance was slightly reduced. How to quantify this? At what gain in computing efficiency is which performance reduction acceptable? Clearly, there will not be one answer for all purposes, but the recent reports on climate change strongly suggest to begin considering such questions.

Quantitative metrics for hypothetical improvements over MSA-based methods? If methods using single sequences without MSAs perform as well as, or even better than, SOTA methods using MSAs, could we quantify metrics measuring the hypothetical improvements from embeddings? This question raised by an anonymous reviewer opens an interesting new perspective. Gain in speed, reduction of computational costs clearly could evolve as one such metric. A related issue is related to protein design: for some applications, the difference in speed might open new doors. Although we have no data to show for others, we could imagine yet another set of metrics measuring the degree to which embedding-based methods realize more protein-specific than family averaged predictions.

Conclusions

Embeddings extracted from protein Language Models (pLMs, Fig. 1), namely from ProtBert and ProtT5 (Elnaggar et al. 2021) and ESM-1b (Rives et al. 2021), contain information that sufficed to predict residue conservation in protein families without using multiple sequence alignments (MSAs, Fig. 3). Such predictions of conservation combined with BLOSUM62 scores predicted the effects of sequence variation (single amino acid variants, or SAVs) without optimizing any additional free parameter (*ProtT5beff*, Fig. 6). Through further training on binary experimental data (effect/neutral), we developed *VESPA*, a relatively simple, yet apparently successful new method for SAV effect prediction (Fig. 4). This method even worked so well on non-binary data from 39 DMS experiments that without ever using such data nor ever using MSAs; VESPA appeared competitive with the SOTA (Fig. 5, Fig. 6), although for SAV effect predictions, embedding-based methods are still not yet outperforming the MSA-based SOTA as for other prediction tasks (Elnaggar et al. 2021; Littmann et al. 2021a, b, c; Stärk et al. 2021). Embedding-based predictions are blazingly fast, thereby they save computing, and ultimately energy resources when applied to daily sequence analysis. In combination, our results suggested that the major signal captured by variant effect predictions originates from some biophysical constraint revealed by raw protein sequences. The ConSurf10k dataset is available at https://doi.org/10.5281/zenodo.5238537. For high-throughput predictions, methods are available through bio_embeddings (Dallago et al. 2021). For single queries VESPA and ProtT5cons will be made available through the PredictProtein server (Bernhofer et al. 2021). VESPA and VESPAl are also available from github at https://github.com/Rostlab/VESPA.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00439-021-02411-y.

Acknowledgements Thanks to Tim Karl and Inga Weise (both TUM) for invaluable help with technical and administrative aspects of this work. Thanks to Nir Ben-Tal (Tel Aviv U) and his team for the excellent services around ConSurf-DB and ConSeq; to Yana Bromberg (Rutgers U), Max Hecht (Amazon) for advancing SNAP; to Adam Riesselman, John Ingraham, and Debbie Marks (Harvard) for making their collection of DMS data available; to Elodie Laine (Sorbonne U) for providing the predictions of GEMME; to the group around Facebook AI Research for making ESM-1b and ESM-1v readily available; to the Dunbrack lab for Pisces, and most importantly to Martin Steinegger (Seoul Natl. Univ.) and his team for MMseqs2 and BFD. Particular thanks to two anonymous reviewers who helped crucially with improving this work and to the valuable comments from the editors. Last, but not least, thanks to all who deposit their experimental data in public databases, and to those who maintain these databases.

Author contributions CM implemented and evaluated the methods and took the lead in writing the manuscript. MH conceived, trained, and evaluated the neural networks on conservation prediction, contributed ideas, and proofread the manuscript. TO and CD contributed crucial ideas and provided valuable comments. MB helped in generating the evaluation methods ConSeq and SNAP2. KE supported the work with coding advice and created the original ConSurf10k data set. DN contributed the clusters and subsets of the SNAP2 development set. BR supervised and guided the work and co-wrote the manuscript. All authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This work was supported by the DFG grant RO 1320/4-1, Software Campus Funding (BMBF 01IS17049) and the KONWIHR Program and by the Bavarian Ministry for Education through funding to the TUM.

Declarations

Conflict of interest No author declares any competing interest.

References

Adzhubei IA, Schmidt S, Peshkin L, Ramensky VE, Gerasimova A, Bork P, Kondrashov AS, Sunyaev SR (2010) A method and server for predicting damaging missense mutations. Nat Methods 7:248–249. https://doi.org/10.1038/nmeth0410-248

Alley EC, Khimulya G, Biswas S, AlQuraishi M, Church GM (2019) Unified rational protein engineering with sequence-based deep representation learning. Nat Methods 16:1315–1322. https://doi.org/10.1038/s41592-019-0598-1

Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res 25:3389–3402. https://doi.org/10.1093/nar/25.17.3389

Alva V, Söding J, Lupas AN (2015) A vocabulary of ancient peptides at the origin of folded proteins. Elife. https://doi.org/10.7554/eLife.09410

Amberger JS, Bocchini CA, Scott AF, Hamosh A (2019) OMIM.org: leveraging knowledge across phenotype-gene relationships. Nucleic Acids Res 47:D1038–D1043. https://doi.org/10.1093/nar/gky1151

AVE Alliance Founding Members (2020) Atlas of Variant Effect Alliance.

Ben Chorin A, Masrati G, Kessel A, Narunsky A, Sprinzak J, Lahav S, Ashkenazy H, Ben-Tal N (2020) ConSurf-DB: An accessible repository for the evolutionary conservation patterns of the majority of PDB proteins. Protein Sci 29:258–267. https://doi.org/10.1002/pro.3779

Ben-Tal N, Lupas AN (2021) Editorial overview: Sequences and topology: 'paths from sequence to structure.' Curr Opin Struct Biol. https://doi.org/10.1016/j.sbi.2021.05.005

Bepler T, Berger B (2019a) Learning protein sequence embeddings using information from structure. arXiv. https://arxiv.org/abs/astro-ph/1902.08661

Bepler T, Berger B (2019b) Learning protein sequence embeddings using information from structure Seventh International Conference on Learning Representations

Bepler T, Berger B (2021) Learning the protein language: evolution, structure, and function. Cell Syst 12(654–669):e3. https://doi.org/10.1016/j.cels.2021.05.017

Berezin C, Glaser F, Rosenberg J, Paz I, Pupko T, Fariselli P, Casadio R, Ben-Tal N (2004) ConSeq: the identification of functionally and structurally important residues in protein sequences. Bioinformatics (oxford, England) 20:1322–1324. https://doi.org/10.1093/bioinformatics/bth070

Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. Nucleic Acids Res 28:235–242. https://doi.org/10.1093/nar/28.1.235

Bernhofer M, Dallago C, Karl T, Satagopam V, Heinzinger M, Littmann M, Olenyi T, Qiu J, Schutze K, Yachdav G, Ashkenazy H, Ben-Tal N, Bromberg Y, Goldberg T, Kajan L, O'Donoghue S, Sander C, Schafferhans A, Schlessinger A, Vriend G, Mirdita M, Gawron P, Gu W, Jarosz Y, Trefois C, Steinegger M, Schneider R, Rost B (2021) PredictProtein—predicting protein structure and function for 29 years. Nucleic Acids Res. https://doi.org/10.1093/nar/gkab354

Bommasani R, Hudson DA, Adeli E, Altman R, Arora S, von Arx S, Bernstein MS, Bohg J, Bosselut A, Brunskill E, Brynjolfsson E, Buch S, Card D, Castellon R, Chatterji N, Chen A, Creel K, Quincy Davis J, Demszky D, Donahue C, Doumbouya M, Durmus E, Ermon S, Etchemendy J, Ethayarajh K, Fei-Fei L, Finn C, Gale T, Gillespie L, Goel K, Goodman N, Grossman S, Guha N, Hashimoto T, Henderson P, Hewitt J, Ho DE, Hong J, Hsu K, Huang J, Icard T, Jain S, Jurafsky D, Kalluri P, Karamcheti S, Keeling G, Khani F, Khattab O, Kohd PW, Krass M, Krishna R, Kuditipudi R, Kumar A, Ladhak F, Lee M, Lee T, Leskovec J, Levent I, Li XL, Li X, Ma T, Malik A, Manning CD, Mirchandani S, Mitchell E, Munyikwa Z, Nair S, Narayan A, Narayanan D, Newman B, Nie A, Niebles JC, Nilforoshan H, Nyarko J, Ogut G, Orr L, Papadimitriou I, Park JS, Piech C, Portelance E, Potts C, Raghunathan A, Reich R, Ren H, Rong F, Roohani Y, Ruiz C, Ryan J, Ré C, Sadigh D, Sagawa S, Santhanam K, Shih A, Srinivasan K, Tamkin A, Taori R, Thomas AW, Tramèr F, Wang RE, Wang W, et al. (2021) On the Opportunities and Risks of Foundation Models. https://arxiv.org/abs/astro-ph/2108.07258

Bromberg Y, Rost B (2007) SNAP: predict effect of non-synonymous polymorphisms on function. Nucleic Acids Res 35:3823–3835

Bromberg Y, Rost B (2008) Comprehensive in silico mutagenesis highlights functionally important residues in proteins. Bioinformatics 24:i207–i212

Bromberg Y, Rost B (2009) Correlating protein function and stability through the analysis of single amino acid substitutions. BMC Bioinformatics 10:S8. https://doi.org/10.1186/1471-2105-10-s8-s8

Burley SK, Berman HM, Bhikadiya C, Bi C, Chen L, Di Costanzo L, Christie C, Dalenberg K, Duarte JM, Dutta S, Feng Z, Ghosh S, Goodsell DS, Green RK, Guranovic V, Guzenko D, Hudson BP, Kalro T, Liang Y, Lowe R, Namkoong H, Peisach E, Periskova I, Prlic A, Randle C, Rose A, Rose P, Sala R, Sekharan M, Shao C, Tan L, Tao YP, Valasatava Y, Voigt M, Westbrook J, Woo J, Yang H, Young J, Zhuravleva M, Zardecki C (2019) RCSB Protein Data Bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. Nucleic Acids Res 47:D464–D474. https://doi.org/10.1093/nar/gky1004

Capriotti E, Fariselli P, Casadio R (2005) I-Mutant2.0: predicting stability changes upon mutation from the protein sequence or structure. Nucleic Acids Res 33:W306–W310. https://doi.org/10.1093/nar/gki375

Dallago C, Schuetze K, Heinzinger M, Olenyi T, Littmann M, Lu AX, Yang KK, Min S, Yoon S, Morton JT, Rost B (2021) Learned embeddings from deep learning to visualize and predict protein sets. Curr Protoc 1:e113. https://doi.org/10.1002/cpz1.113

Devlin J, Chang M-W, Lee K, Toutanova K (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://arxiv.org/abs/astro-ph/1810.04805 [cs]

Efron B, Halloran E, Holmes S (1996) Bootstrap confidence levels for phylogenetic trees. Proc Nat Acad Sci USA 93:13429–13434

Elnaggar A, Heinzinger M, Dallago C, Rehawi G, Wang Y, Jones L, Gibbs T, Feher T, Angerer C, Steinegger M, Bhowmik D, Rost B (2021) ProtTrans: towards cracking the language of life's code through self-supervised learning. Mach Intell 14:30

Esposito D, Weile J, Shendure J, Starita LM, Papenfuss AT, Roth FP, Fowler DM, Rubin AF (2019) MaveDB: an open-source platform to distribute and interpret data from multiplexed assays of variant effect. Genome Biol 20:223. https://doi.org/10.1186/s13059-019-1845-6

Fariselli P, Martelli PL, Savojardo C, Casadio R (2015) INPS: predicting the impact of non-synonymous variations on protein stability from sequence. Bioinformatics 31:2816–2821. https://doi.org/10.1093/bioinformatics/btv291

Findlay GM, Daza RM, Martin B, Zhang MD, Leith AP, Gasperini M, Janizek JD, Huang X, Starita LM, Shendure J (2018) Accurate classification of BRCA1 variants with saturation genome editing. Nature 562:217–222. https://doi.org/10.1038/s41586-018-0461-z

Fowler DM, Fields S (2014) Deep mutational scanning: a new style of protein science. Nat Methods 11:801–807. https://doi.org/10.1038/nmeth.3027

Fu L, Niu B, Zhu Z, Wu S, Li W (2012) CD-HIT: accelerated for clustering the next-generation sequencing data. Bioinformatics 28:3150–3152. https://doi.org/10.1093/bioinformatics/bts565

Fukushima K (1969) Visual feature extraction by a multilayered network of analog threshold elements. IEEE Trans Syst Sci Cybern 5:322–333. https://doi.org/10.1109/TSSC.1969.300225

Gray VE, Hause RJ, Luebeck J, Shendure J, Fowler DM (2018) Quantitative missense variant effect prediction using large-scale mutagenesis data. Cell Syst 6:116-124.e3. https://doi.org/10.1016/j.cels.2017.11.003

Grimm DG, Azencott CA, Aicheler F, Gieraths U, Macarthur DG, Samocha KE, Cooper DN, Stenson PD, Daly MJ, Smoller JW, Duncan LE, Borgwardt KM (2015) The evaluation of tools used to predict the impact of missense variants is hindered by two types of circularity. Hum Mutat 36:513–523. https://doi.org/10.1002/humu.22768

Hecht M, Bromberg Y, Rost B (2013) News from the protein mutability landscape. J Mol Biol 425:3937–3948. https://doi.org/10.1016/j.jmb.2013.07.028

Hecht M, Bromberg Y, Rost B (2015) Better prediction of functional effects for sequence variants. BMC Genomics 16:S1. https://doi.org/10.1186/1471-2164-16-s8-s1

Heinzinger M, Elnaggar A, Wang Y, Dallago C, Nechaev D, Matthes F, Rost B (2019) Modeling aspects of the language of life through transfer-learning protein sequences. BMC Bioinformatics 20:723. https://doi.org/10.1186/s12859-019-3220-8

Henikoff S, Henikoff JG (1992) Amino acid substitution matrices from protein blocks. Proc Natl Acad Sci 89:10915–10919. https://doi.org/10.1073/pnas.89.22.10915

Hopf TA, Ingraham JB, Poelwijk FJ, Scharfe CP, Springer M, Sander C, Marks DS (2017) Mutation effects predicted from sequence co-variation. Nat Biotechnol 35:128–135. https://doi.org/10.1038/nbt.3769

Jumper J, Evans R, Pritzel A, Green T, Figurnov M, Ronneberger O, Tunyasuvunakool K, Bates R, Zidek A, Potapenko A, Bridgland A, Meyer C, Kohl SAA, Ballard AJ, Cowie A, Romera-Paredes B, Nikolov S, Jain R, Adler J, Back T, Petersen S, Reiman D, Clancy E, Zielinski M, Steinegger M, Pacholska M, Berghammer T, Bodenstein S, Silver D, Vinyals O, Senior AW, Kavukcuoglu K, Kohli P, Hassabis D (2021) Highly accurate protein structure prediction with AlphaFold. Nature. https://doi.org/10.1038/s41586-021-03819-2

Katoh K, Standley DM (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. Mol Biol Evol 30:772–780. https://doi.org/10.1093/molbev/mst010

Katsonis P, Lichtarge O (2014) A formal perturbation equation between genotype and phenotype determines the Evolutionary Action of protein-coding variations on fitness. Genome Res 24:2050–2058. https://doi.org/10.1101/gr.176214.114

Kawabata T, Ota M, Nishikawa K (1999) The protein mutant database. Nucleic Acids Res 27:355–357. https://doi.org/10.1093/nar/27.1.355

Kingma DP, Ba J (2014) Adam: a method for stochastic optimization. pp https://arxiv.org/abs/astro-ph/1412.6980

Kircher M, Witten DM, Jain P, O'Roak BJ, Cooper GM, Shendure J (2014) A general framework for estimating the relative pathogenicity of human genetic variants. Nat Genet 46:310–315. https://doi.org/10.1038/ng.2892

Kolodny R (2021) Searching protein space for ancient sub-domain segments. Curr Opin Struct Biol 68:105–112. https://doi.org/10.1016/j.sbi.2020.11.006

Korber B, Fischer WM, Gnanakaran S, Yoon H, Theiler J, Abfalterer W, Hengartner N, Giorgi EE, Bhattacharya T, Foley B, Hastie KM, Parker MD, Partridge DG, Evans CM, Freeman TM, de Silva TI, Angyal A, Brown RL, Carrilero L, Green LR, Groves DC, Johnson KJ, Keeley AJ, Lindsey BB, Parsons PJ, Raza M, Rowland-Jones S, Smith N, Tucker RM, Wang D, Wyles MD, McDanal C, Perez LG, Tang H, Moon-Walker A, Whelan SP, LaBranche CC, Saphire EO, Montefiori DC (2020) Tracking changes in SARS-CoV-2 spike: evidence that D614G increases infectivity of the COVID-19 virus. Cell 182:812-827.e19. https://doi.org/10.1016/j.cell.2020.06.043

Laha S, Chakraborty J, Das S, Manna SK, Biswas S, Chatterjee R (2020) Characterizations of SARS-CoV-2 mutational profile, spike protein stability and viral transmission. Infect Genet Evol 85:104445. https://doi.org/10.1016/j.meegid.2020.104445

Laine E, Karami Y, Carbone A (2019) GEMME: a simple and fast global epistatic model predicting mutational effects. Mol Biol Evol. https://doi.org/10.1093/molbev/msz179

Littmann M, Bordin N, Heinzinger M, Schütze K, Dallago C, Orengo C, Rost B (2021a) Clustering funFams using sequence embeddings improves EC purity. Bioinformatics. https://doi.org/10.1093/bioinformatics/btab371

Littmann M, Heinzinger M, Dallago C, Olenyi T, Rost B (2021b) Embeddings from deep learning transfer GO annotations beyond homology. Sci Rep 11:1160. https://doi.org/10.1038/s41598-020-80786-0

Littmann M, Heinzinger M, Dallago C, Weissenow K, Rost B (2021c) Protein embeddings and deep learning predict binding residues for various ligand classes. bioRxiv. https://doi.org/10.1101/2021.09.03.458869

Liu J, Rost B (2003) Domains, motifs, and clusters in the protein universe. Curr Opin Chem Biol 7:5–11

Liu J, Rost B (2004a) CHOP proteins into structural domain-like fragments. Proteins: structure. Funct Bioinf 55:678–688

Liu J, Rost B (2004b) Sequence-based prediction of protein domains. Nucleic Acids Res 32:3522–3530

Livesey BJ, Marsh JA (2020) Using deep mutational scanning to benchmark variant effect predictors and identify disease mutations. Mol Syst Biol 16:e9380. https://doi.org/10.15252/msb.20199380

Madani A, McCann B, Naik N, Shirish Keskar N, Anand N, Eguchi RR, Huang P, Socher R (2020) ProGen: language modeling for protein generation. arXiv 16:1315

Majithia AR, Tsuda B, Agostini M, Gnanapradeepan K, Rice R, Peloso G, Patel KA, Zhang X, Broekema MF, Patterson N, Duby M, Sharpe T, Kalkhoven E, Rosen ED, Barroso I, Ellard S, UKMD Consortium, Kathiresan S, Myocardial Infarction Genetics, O'Rahilly S, UKCL Consortiun, Chatterjee K, Florez JC, Mikkelsen T, Savage DB, Altshuler D (2016) Prospective functional classification of all possible missense variants in PPARG. Nat Genet 48:1570–1575. https://doi.org/10.1038/ng.3700

Matreyek KA, Starita LM, Stephany JJ, Martin B, Chiasson MA, Gray VE, Kircher M, Khechaduri A, Dines JN, Hause RJ, Bhatia S, Evans WE, Relling MV, Yang W, Shendure J, Fowler DM (2018) Multiplex assessment of protein variant abundance by massively parallel sequencing. Nat Genet 50:874–882. https://doi.org/10.1038/s41588-018-0122-z

Meier J, Rao R, Verkuil R, Liu J, Sercu T, Rives A (2021) Language models enable zero-shot prediction of the effects of mutations on protein function. bioRxiv. https://doi.org/10.1101/2021.07.09.450648

Mercatelli D, Giorgi FM (2020) Geographic and genomic distribution of SARS-CoV-2 mutations. Front Microbiol. https://doi.org/10.3389/fmicb.2020.01800

Miller M, Bromberg Y, Swint-Kruse L (2017) Computational predictors fail to identify amino acid substitution effects at rheostat positions. Sci Rep 7:41329. https://doi.org/10.1038/srep41329

Mistry J, Finn RD, Eddy SR, Bateman A, Punta M (2013) Challenges in homology search: HMMER3 and convergent evolution of coiled-coil regions. Nucleic Acids Res 41:e121. https://doi.org/10.1093/nar/gkt263

Ng PC, Henikoff S (2003) SIFT: predicting amino acid changes that affect protein function. Nucleic Acids Res 31:3812–3814

Niroula A, Urolagin S, Vihinen M (2015) PON-P2: prediction method for fast and reliable identification of harmful variants. PLoS ONE 10:e0117380. https://doi.org/10.1371/journal.pone.0117380

Nishikawa K, Ishino S, Takenaka H, Norioka N, Hirai T, Yao T, Seto Y (1994) Constructing a protein mutant database. Protein Eng 7:733. https://doi.org/10.1093/protein/7.5.733

O'Donoghue SI, Schafferhans A, Sikta N, Stolte C, Kaur S, Ho BK, Anderson S, Procter J, Dallago C, Bordin N, Adcock M, Rost B (2020) SARS-CoV-2 structural coverage map reveals state changes that disrupt host immunity. bioRxiv. https://doi.org/10.1101/2020.07.16.207308

Ofer D, Brandes N, Linial M (2021) The language of proteins: NLP, machine learning and protein sequences. Comput Struct Biotechnol J 19:1750–1758. https://doi.org/10.1016/j.csbj.2021.03.022

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825–2830

Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I (2019) Language Models are Unsupervised Multitask Learners. 24.

Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu PJ (2020) Exploring the limits of transfer learning with a unified text-to-text transformer. https://arxiv.org/abs/astro-ph/1910.10683[cs, stat].

Ramensky V, Bork P, Sunyaev S (2002) Human non-synonymous SNPs: server and survey. Nucleic Acids Res 30:3894–3900

Rao R, Meier J, Sercu T, Ovchinnikov S, Rives A (2020) Transformer protein language models are unsupervised structure learners. bioRxiv. https://doi.org/10.1101/2020.12.15.422761

Reeb J (2020) Data for: Variant effect predictions capture some aspects of deep mutational scanning experiments. 1. doi: https://doi.org/10.17632/2rwrkp7mfk.1

Reeb J, Hecht M, Mahlich Y, Bromberg Y, Rost B (2016) Predicted molecular effects of sequence variants link to system level of disease. PLoS Comput Biol 12:e1005047. https://doi.org/10.1371/journal.pcbi.1005047

Reeb J, Wirth T, Rost B (2020) Variant effect predictions capture some aspects of deep mutational scanning experiments. BMC Bioinf 21:107. https://doi.org/10.1186/s12859-020-3439-4

Riesselman AJ, Ingraham JB, Marks DS (2018) Deep generative models of genetic variation capture the effects of mutations. Nat Methods 15:816–822. https://doi.org/10.1038/s41592-018-0138-4

Rives A, Meier J, Sercu T, Goyal S, Lin Z, Liu J, Guo D, Ott M, Zitnick CL, Ma J, Fergus R (2021) Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. Proc Natl Acad Sci. https://doi.org/10.1073/pnas.2016239118

Rost B (1996) PHD: predicting one-dimensional protein structure by profile based neural networks. Methods Enzymol 266:525–539

Rost B, Sander C (1992) Jury returns on structure prediction. Nature 360:540

Rost B, Sander C (1993) Prediction of protein secondary structure at better than 70% accuracy. J Mol Biol 232:584–599. https://doi.org/10.1006/jmbi.1993.1413

Schelling M, Hopf TA, Rost B (2018) Evolutionary couplings and sequence variation effect predict protein binding sites. Proteins 86:1064–1074. https://doi.org/10.1002/prot.25585

Schwarz JM, Rodelsperger C, Schuelke M, Seelow D (2010) MutationTaster evaluates disease-causing potential of sequence alterations. Nat Methods 7:575–576. https://doi.org/10.1038/nmeth0810-575

Sim N-L, Kumar P, Hu J, Henikoff S, Schneider G, Ng PC (2012) SIFT web server: predicting effects of amino acid substitutions on proteins. Nucleic Acids Res 40:W452–W457. https://doi.org/10.1093/nar/gks539

Sruthi CK, Balaram H, Prakash MK (2020) Toward developing intuitive rules for protein variant effect prediction using deep mutational scanning data. ACS Omega 5:29667–29677. https://doi.org/10.1021/acsomega.0c02402

Stärk H, Dallago C, Heinzinger M, Rost B (2021) Light attention predicts protein location from the language of life. bioRxiv. https://doi.org/10.1101/2021.04.25.441334

Steinegger M, Söding J (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nat Biotechnol 35:1026

Steinegger M, Söding J (2018) Clustering huge protein sequence sets in linear time. Nat Commun 9:2542. https://doi.org/10.1038/s41467-018-04964-5

Studer RA, Dessailly BH, Orengo CA (2013) Residue mutations and their impact on protein structure and function: detecting beneficial and pathogenic changes. Biochem J 449:581–594. https://doi.org/10.1042/BJ20121221

Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. Proceedings of the 27th International Conference on Neural Information Processing Systems—Volume 2. MIT Press, Montreal, Canada, pp 3104–3112

The UniProt Consortium (2021) UniProt: the universal protein knowledgebase in 2021. Nucleic Acids Res 49:D480–D489. https://doi.org/10.1093/nar/gkaa1100

Wang G, Dunbrack RL Jr (2003) PISCES: a protein sequence culling server. Bioinformatics 19:1589–1591. https://doi.org/10.1093/bioinformatics/btg224

Wang Z, Moult J (2001) SNPs, protein structure, and disease. Hum Mutat 17:263–270. https://doi.org/10.1002/humu.22

Weile J, Roth FP (2018) Multiplexed assays of variant effects contribute to a growing genotype–phenotype atlas. Hum Genet 137:665–678. https://doi.org/10.1007/s00439-018-1916-x

Weißenow K, Heinzinger M, Rost B (2021) Protein language model embeddings for fast, accurate, alignment-free protein structure prediction. bioRxiv. https://doi.org/10.1101/2021.07.31.454572

Zhou G, Chen M, Ju CJT, Wang Z, Jiang JY, Wang W (2020) Mutation effect estimation on protein-protein interactions using deep contextualized representation learning. NAR Genom Bioinform. https://doi.org/10.1093/nargab/lqaa015

A.1.4. Heinzinger *et al.*, NAR Genomics and Bioinformatics (2022)

https://doi.org/10.1093/nargab/lqac043

Contrastive learning on protein embeddings enlightens midnight zone

Michael Heinzinger[1,2,*], Maria Littmann[1], Ian Sillitoe[3], Nicola Bordin[3], Christine Orengo[3] and Burkhard Rost[1,4,*]

[1]TUM (Technical University of Munich) Dept Informatics, Bioinformatics & Computational Biology - i12, Boltzmannstr. 3, 85748 Garching/Munich, Germany, [2]TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Boltzmannstr. 11, 85748 Garching, Germany, [3]Institute of Structural and Molecular Biology, University College London, London WC1E 6BT, UK and [4]Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, 85748 Garching, Germany & TUM School of Life Sciences Weihenstephan (WZW), Alte Akademie 8, Freising, Germany

Received November 22, 2021; Revised March 25, 2022; Editorial Decision May 09, 2022; Accepted May 17, 2022

ABSTRACT

Experimental structures are leveraged through multiple sequence alignments, or more generally through homology-based inference (HBI), facilitating the transfer of information from a protein with known annotation to a query without any annotation. A recent alternative expands the concept of HBI from sequence-distance lookup to embedding-based annotation transfer (EAT). These embeddings are derived from protein Language Models (pLMs). Here, we introduce using single protein representations from pLMs for contrastive learning. This learning procedure creates a new set of embeddings that optimizes constraints captured by hierarchical classifications of protein 3D structures defined by the CATH resource. The approach, dubbed *ProtTucker*, has an improved ability to recognize distant homologous relationships than more traditional techniques such as threading or fold recognition. Thus, these embeddings have allowed sequence comparison to step into the 'midnight zone' of protein similarity, i.e. the region in which distantly related sequences have a seemingly random pairwise sequence similarity. The novelty of this work is in the particular combination of tools and sampling techniques that ascertained good performance comparable or better to existing state-of-the-art sequence comparison methods. Additionally, since this method does not need to generate alignments it is also orders of magnitudes faster. The code is available at https://github.com/Rostlab/EAT.

INTRODUCTION

Phase-transition from daylight through twilight into midnight zone

Protein sequence determines structure which determines function. This simple chain underlies the success of grouping proteins into families from sequence (1–4). Information from experimental high-resolution three-dimensional (3D) structures expands the perspective from families to superfamilies (5,6) that often reveal evolutionary and functional connections not recognizable from sequence alone (7,8). Thus, 3D information helps us to penetrate through the twilight zone of sequence alignments (9,10) into the midnight zone of distant evolutionary relationships (11).

The transition from daylight, through twilight and into the midnight zone is characterized by a phase-transition, i.e. a sigmoid function describing an order of magnitude increase in recall (relations identified) at the expense of a decrease in precision (relations identified correctly) over a narrow range of sequence similarity. Measuring sequence similarity by the HSSP-value (HVAL) (10,12) for the *daylight zone* at HVAL $>$ 5 ($>$25% PIDE - pairwise sequence identity over $>$250 aligned residues) over 90% of all protein pairs have similar 3D structures, while at the beginning of the *midnight zone* for HVAL$\leftarrow$5 ($<$15% PIDE for $>$ 250 aligned residues), over 90% have different 3D structures. Thus, the transition from daylight to midnight zone is described by a phase-transition in which over about ten percentage points in PIDE precision drops from 90% to 10%, i.e. from almost *all correct* to almost *all incorrect* within ±5 points PIDE. The particular point at which the twilight zone begins and how extreme the transition is, depends on the phenotype: steeper at lower PIDE for structure (10) and flatter at higher PIDE for function (13,14).

[*]To whom correspondence should be addressed. Email: mheinzinger@rostlab.org
Correspondence may also be addressed to Burkhard Rost. Tel: +49 289 17 811; Email: assistant@rostlab.org

If two proteins have highly similar structures, it is still possible for their sequences to be found in this *midnight zone*, i.e. have seemingly random sequence similarity (11). Thus, if we could safely lower the threshold just a little, we would gain many annotations of structural and functional similarity. In fact, any push a little lower reveals many proteins with similar phenotype, e.g. structure or function. Unfortunately, without improving the search method, such a lowering usually comes at the expense of even more proteins with dissimilar phenotype.

This simple reality has been driving the advance of methods using sequence similarity to establish relations: from advanced pairwise comparisons (15,16) over sequence-profile (17–20) to profile-profile comparisons (8,21,22,23,24,25,26) or efficient shortcuts to the latter (27,28). All those methods share one simple idea, namely, to use evolutionary information (EI) to create families of related proteins. These are summarized in multiple sequence alignments (MSAs). Using such information as input to machine learning methods has been generating essentially all state-of-the-art (SOTA) prediction methods for almost three decades (29–31). Using MSAs has also been one major key behind the breakthrough in protein structure prediction through *AlphaFold2* (32), and subsequently of *RoseTTAFold* (33) which builds on ideas introduced by *AlphaFold2*, i.e. allowing for communication between different sequence- and structure modules within the network. Transfer- or representation-learning offer a novel route toward comparisons of and predictions for single sequences without MSAs.

Embeddings capture language of life written in proteins

The introduction of LSTM- or attention-based Language Models (LMs) such as ELMo (34) or BERT (35) enabled a better use of large, unlabeled text corpora which arguably improved all tasks in natural language processing (NLP) (36). These advances have been transferred to proteins through protein Language Models (pLMs) equating amino acids with words in NLP and the sequence of entire proteins with sentences. Such pLMs learn to predict masked or missing amino acids using large databases of raw protein sequences as input (37–43), or by refining the pLM through another supervised task (44,45). Processing the information learned by the pLM, e.g. by using the output of the last hidden layers of the networks forming the pLMs, yields a representation of protein sequences referred to as embeddings (Figure 1 in (37)). Embeddings have been used successfully as exclusive input to predicting secondary structure and subcellular localization at performance levels almost reaching (38–40) or even exceeding (37,46,47) the SOTA using evolutionary information from MSAs as input. Embeddings can even substitute sequence similarity for homology-based annotation transfer (48,49). The power of such embeddings has been increasing with the advance of algorithms and the growth of data (37). The recent advances have shown that a limit to such improvements has not nearly been reached when writing this (22.02.2022).

Embeddings from pLMs capture a diversity of higher-level features of proteins, including various aspects of protein function and structure (37,38,40,48,49,50,51). In fact, pLMs such as ProtT5 (37) or ESM-1b (38) capture aspects about protein structure so impressively that inter-residue distances – and consequently 3D structure – can be predicted without using MSAs, even with relatively small (few free parameters) Deep Learning (DL) architectures (52).

Supervised learning directly maps the input to the class output. Instead, contrastive learning (53), optimizes a new embedding space in which similar samples are pushed closer, dissimilar samples farther apart. Contrastive learning relies only on the similarity between pairs (or triplets) of samples instead of on class label. The definition of similarity in embedding rather than sequence space, combined with contrastive learning, offered an alternative to sequence-based protein comparisons. This led us to hypothesize that we might find structurally and functionally consistent subgroups within protein families from raw sequences. As a proof-of-principle, a rudimentary precursor of this work helped to cluster FunFams (54,49). The benefit of optimizing embeddings specifically for SCOPe fold recognition (55) has recently been shown (44,50,56). Other approaches toward fold recognition deep learn fold-specific motifs (57), pairwise similarity scores (58) or sequence alignments (59). However, most of the top-performing solutions rely on information extracted from MSAs (60) and do not utilize the transfer-learning capabilities offered by recent pLMs.

Here, we expand on the hypothesis that replacing supervised learning by contrastive learning intrinsically fits the hierarchy of CATH (5,54). We propose an approach that marries both, self-supervised pretraining and contrastive learning, by representing protein sequences as embeddings, and using increasing overlap in the CATH hierarchy as a notion of increasing structural similarity to contrastively learn a new embedding space. We used the pLM ProtT5 (37) as static feature encoder (no fine-tuning of the pLM) to retrieve initial embeddings that were then mapped by a feed-forward neural network (FNN) to a new, learned embedding space optimized on CATH through contrastive learning. More specifically, the Soft Margin Loss was used with triplets of proteins (anchor, positive, and negative) to optimize the new embedding space toward maximizing the distance between proteins from different CATH classes (anchor-negative pairs) while minimizing the distance between proteins in the same CATH class (anchor-positive pairs). Triplets of varying structural similarity were used simultaneously to optimize a single, shared network: all four CATH-levels were simultaneously learned by one FNN. The resulting model was called *ProtTucker* and its embeddings were established to identify more distant relations than is possible from sequence alone. One important objective of *ProtTucker* is to study entire functional modules through identifying more distant relations, as found to be crucial for capturing mimicry and hijacking of SARS-CoV-2 (61).

METHODS

CATH hierarchy

The CATH (6,54) hierarchy (v4.3) classifies three-dimensional (3D) protein structures from the PDB (Protein Data Bank (62)) at the four levels Class, Architecture, Topology and Homologous superfamily. On average, higher levels (further away from root: H > T > A > C) are

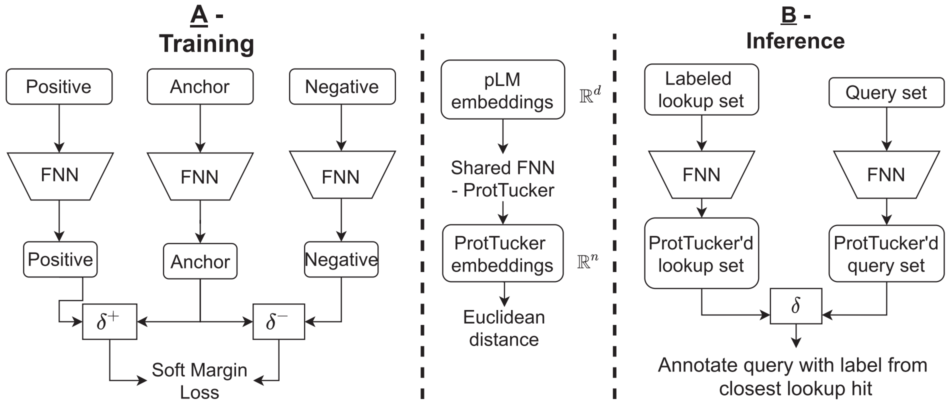

Figure 1. Sketch of ProtTucker. Panel **A** illustrates how protein triplets were used to contrastively learn the CATH hierarchy (5,54). First, protein Language Models (pLMs) were used as static feature encoders to derive embeddings for protein sequences (anchor, positive, negative). The embedding of each protein was processed separately by the same, shared FNN with hard parameter sharing, called ProtTucker. During optimization, the Soft Margin Loss was used to maximize the distance between proteins from different CATH classes (anchor-negative pairs) while minimizing the distance between proteins in the same CATH class (anchor-positive pairs). All four CATH-levels were simultaneously learned by the same FNN. This resulted in a newly, learned CATH-optimized embedding for each protein. Panel **B** sketches how the contrastive learning FNN is used for prediction of new proteins (inference). For all proteins in a lookup set with experimental annotations (labeled proteins; here the CATH lookup set), as well as for a query protein without experimental annotations (unlabeled proteins) all embeddings are extracted in two steps: (1) extract per-residue embeddings from original pLM and create per-protein embeddings by averaging over protein length. (2) Input those embeddings into the pre-trained FNNs, i.e. ProtTucker. Similar to homology-based inference (HBI), predictions are generated by transferring the annotation of the closest hit from the lookup set to the query protein. The embedding-based annotation transfer (EAT) transferred annotations to the hit with the smallest Euclidean distance in *ProtTucker* embedding space.

more similar in their 3D structure or have more residues for which the same level of 3D similarity is reached. We used increasing overlap in this hierarchical classification as a proxy to define increasing structural similarity between protein pairs. For example, we assumed that any two proteins with the same topology (T) are structurally more similar than any two proteins with identical architecture (A) but different topology (T). In more formal terms: SIM3D(P1,P2)>SIM3D(P3,P4), where T(P1) = T(P2) & T(P3)≠T(P4) & A(P3) = A(P4). This notion of similarity was applied on all four levels of CATH.

Data set

The sequence-unique datasets provided by CATH (5,54) v4.3 (123k proteins, CATH-S100) provided training and evaluation data for ProtTucker. A test set (300 proteins, dubbed test300 in the following) for final evaluation and a validation set (200 proteins, dubbed *val200*) for early stopping were randomly split off from CATH-S100 while ensuring that (1) every homologous superfamily appeared maximally once in test300 ∩ val200 and (2) each protein in test300 & val200 has a so called Structural Sub-group (SSG) annotation, i.e. clusters of domain structure relatives that superpose within 5Å (0.5 nm), in CATH. To create the training set, we removed any protein from CATH-S100 that shared more than 20% pairwise sequence identity (PIDE) to any validation or test protein according to MMSeqs2 (27) applying its iterative profile-search (*–num-iterations 3*) with highest sensitivity (*-s 7.5*) and bidirectional coverage (*–cov-mode 0*). Additionally, large families (>100 members) within CATH-S100 were clustered at 95% PIDE and length coverage of 95% of both proteins using MMSeqs2 (bidirectional coverage; *–cov-mode 0*). The cluster representatives were used for training (66k proteins, dubbed *train66k*) and as lookup set during early stopping on set *val200*. We needed a lookup set from which to transfer annotations because contrastive learning outputs embeddings instead of class predictions. For the final evaluation on *test300*, we created another lookup set but ignored val200 proteins during redundancy reduction (69k proteins, dubbed *lookup69k*). This provided a set of proteins for validation which had similar sequence properties to those during the final evaluation while 'hiding' them during training and not using them for any other optimization. To ensure strict non-redundancy between lookup69k and test300, we further removed any protein from *test300* with HVAL > 0 (10) to any protein in *lookup69k* (219 proteins, dubbed *test219* in the following). All performance measures were computed using *test219*.

Data augmentation can be crucial for contrastive learning to reach performance in other fields (63). However, no straightforward way exists to augment protein sequences as randomly changing sequences very likely alters or even destroys protein structure and function. Therefore, we decided to use homology-based inference (HBI) for data augmentation during training, i.e. we created a new training set based on Gene3D (64) (v21.0.1) which uses Hidden Markov Models (HMMs) derived from CATH domain structures to transfer annotations from labeled CATH to unlabeled UniProt. We first clustered the 61M protein sequences in Gene3D at 50% PIDE and 80% coverage of both proteins (bidirectional coverage; *–cov-mode 0*) and then applied the same MMSeqs2 profile-search (*–num-iterations 3 –s 7.5*) as outlined above to remove cluster representatives with ≥ 20% PIDE to any protein in test300 or val200 (PIDE(P_{train},$P_{\text{test300|val200}}$) ≤ 20%). This filtering yielded 11M sequences for an alternative training set (dubbed *train11M*).

The CATH detection of ProtTucker was further analyzed using a strictly non-redundant, high-quality dataset. This set was created by first clustering CATH v4.3 at 30% using HMM profiles from HMMER and additionally discarding all proteins without equivalent entry in SCOPe, i.e. the domain boundaries and the domain-superfamily assignment had to be nearly identical (3186 proteins, CATH-S30). We used the highly sensitive structural alignment scoring tool SSAP (65,66) to compute the structural similarity between all protein pairs in this set.

We probed whether ProtTucker embeddings might also help in solving tasks unrelated to protein structure/CATH, using as proxy a dataset assessing subcellular location prediction in ten states (46,67). We embedding-transferred annotations (EAT) from the standard *DeepLocTrain* set to 490 proteins in a recently proposed test set (*setHard*) that was strictly non-redundant to *DeepLocTrain*. Datasets described elsewhere in more detail (46,67). Finally, we showcased predictions for entire organisms using three UniProt reference proteome: *Escherichia coli* (E. Coli; reviewed, Swiss-Prot (68)), *Armillaria ostoyae* (A. ostoyae; unreviewed, TrEMBL (68)) and *Megavirus Chilensis* (M. Chilensis; unreviewed TrEMBL (68)).

Data representation

Protein sequences were encoded through distributed vector representations (embeddings) derived from four different pre-trained protein language models (pLM): (–) **ProtBERT** (37) based on the NLP (Natural Language Processing) algorithm BERT (35) but trained on BFD (Big Fantastic Database) with over 2.3 billion protein sequences (69). (2) **ESM-1b** (38) is similar to (Prot)BERT but trained on UniRef50 (68). (3) **ProtT5-XL-U50** (37) (*ProtT5* for simplicity) based on the NLP sequence-to-sequence model T5 (70) trained on BFD and fine-tuned on Uniref50. (4) **ProSE** (44) trained long short-term memory cells (LSTMs) either solely on 76M unlabeled sequences from UniRef90 (**ProSE-DLM**) or on additionally predicting intra-residue contacts and structural similarity from 28k SCOPe proteins (55) (multi-task: **ProSE-MT**). While ProSE, ProtBert and ESM-1b were trained on reconstructing corrupted tokens/amino acids from non-corrupted (protein) sequence context (masked language modeling), ProtT5 was trained by *teacher forcing*, i.e. input and targets were fed to the model with inputs being corrupted protein sequences and targets being identical to inputs but shifted to the right (span generation with span size of 1 for ProtT5). Except for ProSE-MT, all pLMs were optimized only through self-supervised learning exclusively using unlabeled sequences for pre-training.

pLMs output a single vector for each residue yielding an $L \times N$-dimensional matrix (L: protein length, N: embedding dimension; $N = 1024$ for ProtBERT/ProtT5; $N = 1280$ for ESM-1b; $N = 6165$ for ProSE). From this $L \times N$ embedding matrix, we derived a fixed-size N-dimensional vector representing each protein by averaging over protein length (Figure 1 (37)). The pLMs were used as static feature encoder only, i.e. no gradient was backpropagated for fine-tuning. As recommended in the original publication (37), for ProtT5, we only used the encoder part of ProtT5 in half-precision to embed protein sequences. Similarly, ProtBERT embeddings were derived in half-precision.

Contrastive learning: architecture

A two-layer feedforward neural network (FNN) projected fixed-size per-protein (sentence-level) embed-dings from 1024-d (1280-d/6165-d for ESM-1b and ProSE respectively) to 256 and further to 128 dimensions with the standard hyperbolic tangent (*tanh*) as non-linearity between layers. We also experimented with deeper/more sophisticated networks without any gain from more free parameters (data not shown). This confirmed previous findings that simple networks suffice when inputting advanced embeddings (37,38,52,71). As the network was trained using contrastive learning, no final classification layer was needed. Instead, the 128-dimensional output space was optimized directly.

Contrastive learning: training

During training, the new embedding space spanned by the output of the FNN was optimized to capture structural similarity using triplets of protein embeddings. Each triplet had an anchor, a positive and a negative. In each epoch, all *train66k* proteins were used once as anchor, while positives and negatives were sampled randomly from *train66k* using the following *hierarchy-sampling*. First, a random level α (α = [1,2,3,4]) describing the increasing structural overlap between triplets was picked. This defined a positive (same CATH-number as anchor up to level α'≤α) and a negative label (same CATH-number as anchor up to level α'< α). For instance, assume the anchor has the CATH-label 1.25.10.60 (Rad61, Wapl domain) and we randomly picked α = 3 (topology-level), only proteins with the anchor's topology (1.25.10.x; Leucine-rich Repeat Variant) qualify as positives while all negatives share the anchor's architecture (1.25.y.z; Alpha Horseshoe) with different topology (y≠10). Self-hits of the anchor were excluded. From the training proteins fulfilling those constraints, one positive and one negative were picked at random. If no triplets could be formed (e.g. α = 4 with a single-member homologous superfamily had no positive for this anchor/α combination), α was changed at random until a valid triplet could be formed (eventually, all proteins found a class-level partner). This flexibility in sampling enabled training on all proteins, independent of family size.

Unlike randomly sampling negatives, the hierarchical sampling could be described as semi-hard sampling as it zoomed into triplets that were neither too easy (little signal) nor too hard (outliers) to classify by ensuring a minimal overlap between the anchor and the chosen negative/positive pair. Thereby, trivial triplets are undersampled (avoided), i.e. those with 3D structures so different that the separation becomes trivial (*daylight zone*). As the final triplet selection was still random, anchor-positive pairs could still be too easy/similar which was shown to hinder the success of contrastive learning (72). To solve this issue, we paired hierarchy-sampling with so called *batch-hard* *sampling* (72) which offers a computationally efficient solution for sampling *semi-hard* triplets within one mini-batch. More specifically, we combined *batch-hard sampling* with the triplets created using *hierarchy-sampling* by re-wiring all proteins, irrespective of anchor, positive or negative, within one mini-batch such that they satisfied the hierarchy-sampling criterion and had maximum/minimal Euclidean distance for anchor-positive/anchor-negative pairs. Sampling hard triplets only within each mini-batch instead of across the entire data set avoided extreme outliers (potentially too hard/noisy) while increasing the rate of semi-hard anchor-positive/anchor-negative pairs. Assume multiple proteins of the topology Leucine-rich Repeat Variant were within one mini-batch, the hardest positive for each anchor would be picked by choosing the anchor-positive pair with the largest Euclidean distance. Accordingly, anchor-negative pairs would be picked based on the smallest Euclidean distance. For each mini-batch, this sampling was applied to all four levels of the CATH-hierarchy, so triplets were re-wired on all four CATH levels resulting in a total batch-size of about: batch_size * 3 * 4. This was an 'about' instead of 'equal' because for some mini-batches, not all proteins had valid triplets for all four levels.

Finally, the same two-layer FNN was used (hard parameter sharing) to project the 1024-d (or 1280-d/6165-d for ESM-1b or ProSE respectively) embeddings of all proteins, irrespective of anchor, positive or negative, to a new 128-d vector space. The *Soft Margin Loss* was used to optimize this new embedding space such that anchor-positive pairs were pulled together (reduction of Euclidean distance) while pushing apart anchor-negative pairs (increase of Euclidean distance). The efficiency of combining the Soft Margin Loss with batch-hard sampling was established before (72), although without prior hierarchical triplet sampling. Here, we used Soft Margin Loss as implemented in PyTorch:

$$Loss\ (d, y) = \sum_{t} \frac{\log\left(1 + e^{(-y[t]*d[t])}\right)}{|d|} \qquad (1)$$

$$d = \left\{ \min_{\substack{n \in B \wedge a \neq n \\ C_i(a) \neq C_i(n)}} D(f(a), f(n)) - \max_{\substack{n \in B\ p \neq a \\ C_i(a) \neq C_i(p)}} D(f(a), f(p)) \right\} \qquad (2)$$

$$\forall a \in \{B\} \wedge \forall i \in 1, 2, 3, 4 \wedge \forall C_i \in \{CATH labels\}$$

B represents one mini-batch created through hierarchical sampling, $f(a)$, $f(p)$ and $f(n)$ represent the *ProtTucker* embeddings of proteins a (anchor), n (negative), and p (positive) represented as pLM embeddings; C_i represents the CATH annotation of a protein on the *i*'th hierarchy level of CATH; finally, D(f(a),f(x)) represents the Euclidean distance between the embeddings for proteins a and x. We created the mini-batch B used for training by choosing for each protein or anchor a in B the hardest negative n and the hardest positive p by picking those proteins in B that have the smallest | largest Euclidean distance D to a *ProtTucker* embedding space while not sharing | sharing C_i, respectively. Those semi-hard triplets are indexed by t and $d[t]$ referring to the difference between D of anchor-negative and D of anchor-positive. In our case, the label for the *t*'th triplet $y[t]$ is always 1 as the sign of x indicates training success, i.e. whether the distance anchor-positive is smaller than that between anchor-negative.

Consequently, triplets of varying structural similarity were used simultaneously to optimize a single, shared network, i.e. all four CATH-levels were learned by the same network at the same time (Figure 1A). We used the *Adam optimizer* (73) with a learning rate of 0.001, and a batch-size of 256 to optimize the network. The effective batch-size increased due to batch-hard sampling to a maximum of 3072, depending on the number of valid triplets that could be formed within the current mini-batch. Training terminated (*early stopping*) at the highest accuracy in predicting the correct homologous superfamily for set *val200*.

Evaluation and prediction (inference)

While supervised training directly outputs class predictions, contrastive learning, outputs a new embedding space. Thus, predictions (inferences) were generated as for homology-based inference (HBI), i.e. given protein X with experimental annotation (CATH assignment) and a query protein Q without, then HBI transfers the annotation from X to Q if sequence similarity SIM(X,Q)> threshold. For contrastive learning, we replaced SIM(X,Q) by D(f(X),f(Q)) with D as the shortest Euclidean distance in embedding space (Figure 1B). In previous studies (37,48,49), we found the Euclidean distance performed better than the cosine distance which is more common in AI/NLP. The Euclidean distance also optimized the ProtTucker embeddings. Set *test219* with the *lookup69k* as lookup set (set of all X) served as final evaluation. If no protein in the lookup set shared the annotation of the query protein at a certain CATH-level (more likely for H than for C), the sample was excluded from the evaluation of this CATH-level as no correct prediction was possible (Supplementary Table S1).

During evaluation, we compared the performance of our embedding-based annotation transfer (EAT) to HBI using the sequence comparisons from MMSeqs2 (27). While transferring only the HBI hit with the lowest E-value, we searched for hits up to an E-value of 10 to ensure that most proteins had at least one hit while using the highest sensitivity setting (-s 7.5). Additionally, we used publicly available CATH-Gene3D (54) hidden Markov Models (HMMs) along with HMMER (74) to detect remote homologs up to an *E*-value of 10.

For both approaches, EAT and HBI, we computed the accuracy as the fraction of correct hits for each CATH-level. A hit at lower CATH-levels could be correct if and only if all previous levels were correctly predicted. Due to varying number of samples at different CATH-levels (Supplementary Table S1), performance measures not normalized by background numbers could be higher for lower levels. Predictions were counted as incorrect if a query did not have a hit in the lookup set but a lookup protein of the same CATH annotation existed. This not only affected the number of proteins available at different CATH-levels (Supplementary Table S2) but also the number of classes (Supplementary Table S3). A random baseline was computed by transferring annotations from a randomly picked protein in *lookup69k* to *test219*.

Performance measures

The four coarse-grained classes at the top CATH level (‘C’) are defined by their secondary structure content. These four branch into 5481 different superfamilies with distinct structural and functional aspects (CATH v4.3.0). However, most standard metrics are defined for binary cases which requires some grouping of predictions into four cases: 1) TP (true positives): correctly predicted to be in the *positive* class, 2) TN (true negatives): correctly predicted to be in the *negative* class, 3) FP (false positives): incorrectly predicted to be positives, and 4) FN (false negatives): incorrectly predicted to be in in the negative class. Here, we focused on performance measures applicable for multiclass problems and are implemented in scikit (75). These were in particular: **accuracy** (Acc, Equation 3) as the fraction of correct predictions

$$Accuracy\ (y, \hat{y}) = \frac{1}{n\_samples} \sum_{i\,=\,0}^{n\_samples-1} 1\,(\,\widehat{y_i} = y_i\,)\ (3)$$

with y_i being the ground truth (experimental annotation) and $\widehat{y_i}$ the prediction for protein i. In analogy, we defined **coverage** as the proportion of the *test219* proteins for which a classifier made a prediction at a given prediction reliability $\widehat{y_i^r}$ and reliability threshold θ:

$$Coverage\,(y, \hat{y}) \ = \ \frac{1}{n\_samples} \sum_{i\,=\,0}^{n\_samples-1} 1(\hat{y}_i^r\ lt;\theta)\ (4)$$

In these definitions accuracy corresponds to precision, and coverage to recall binarizing a multiclass problem through micro-averaging, i.e. by counting the total TPs, FPs and FNs globally, irrespective of the class. The multi-class extension of Matthew’s correlation coefficient (MCC, (31)) was defined as:

$$MCC \ = \ \frac{c \times s - \sum_k^K p_k \times t_k}{\sqrt{\left(s^2 - \sum_k^K p_k^2\right) \times \left(s^2 - \sum_k^K t_k^2\right)}} \qquad (5)$$

with $t_k = \sum_i^K C_{ik}$ as the number of times class k truly occurred, $p_k = \sum_i^K C_{ki}$ as the number of times class k was predicted, $c \ = \sum_k^K C_{kk}$, the total number of samples correctly predicted, and $s \ = \sum_i^K \sum_j^K C_{ij}$, the total number of samples.

95% confidence intervals for accuracy and MCC were estimated over $n = 1000$ bootstrap sets; for each bootstrap set we randomly sampled predictions from the original test set with replacement. Standard deviation (or in the case of bootstrapping: bootstrap standard error) was calculated as the difference of each test set (x_i) from the average performance $\langle X \rangle$ (Equation 6). 95% confidence intervals were estimated by multiplying the bootstrap standard error by 1.96.

$$\text{StdDev} \ = \sqrt{\frac{x_i - \langle X \rangle^2}{n}} \qquad (6)$$

RESULTS

Generalization of HBI to EAT

Homology-based inference (HBI) uses sequence similarity to transfer annotations from experimentally characterized (labelled) to uncharacterized (unlabeled) proteins. More specifically, an unlabeled query protein Q is aligned against a set of proteins X with experimental annotations (dubbed *lookup set*) and the annotation of the best hit, e.g. measured as lowest E-value, is transferred if it is below a certain threshold (e.g. E-value(Q,X)<10–3). This relates to inferring the annotation of a query protein from the *k*-Nearest Neighbors (k-NN) in sequence space. More recently, similar approaches expanded from distance in sequence to distance in embedding space (Figure 1B) as means for k-NN based annotation transfer (48,50). Here, we refer to such methods as embedding- based annotation transfer (EAT). We used EAT as a proxy for the comparison of embeddings from five different pLMs: ProSE-DLM & ProSE-MT (44), ProtBERT & ProtT5 (37), and ESM-1b (38). Next, we used triplets of proteins (anchor, positive, negative) to learn a new embedding space by pulling protein pairs from the same CATH class (anchor-positive) closer together while pushing apart pairs from different CATH classes (anchor-negative; Figure 1A). We referred to this method as *ProtTucker*. At this stage, we did not fine-tune the pre-trained pLMs. Instead, we created a new embedding space using a two-layer feed-forward neural network (FNN).

EAT with raw embeddings level with HBI

First, we transferred annotations from all proteins in *lookup69k* to any protein in *test219*. All pLMs significantly (at 95% CI—confidence interval) outperformed random annotation transfer (Table 1). Performance differed between pLMs (Table 1), with ProtBERT (37) being consistently worse than LSTM-based ProSE-DLM or more advanced transformers (ESM-1b, ProtT5). ESM-1b and ProtT5 also numerically outperformed ProSE-DLM and HBI using MMseqs2 (27), especially on the most fine-grained and thus hardest level of superfamilies. However, MMseqs2 had been used for redundancy-reduction, i.e. the data set had been optimized for minimal performance of MMseqs2. HBI using publicly available HMM-profiles from CATH-Gene3D (54) along with the profile-based advanced HMMER (74) designed for more remote homology detection, outperformed all raw embeddings for homologous superfamilies, while embeddings from ESM-1b and ProtT5 appeared superior on the class- and architecture-level (Table 1). In fact, all HBI values, for MMseqs2 and HMMER, were highest for the H-level, and second highest for the C-level. In contrast, raw pLM embeddings mirrored the random baseline trend, with numbers being inversely proportional to the rank in C, A, T, H (highest for C, lowest for H, Table 1).

Table 1. Accuracy for annotation transfer to queries in test219 *

	Method/Input	C	A	T	H	Mean
Baseline	Random	29 ± 6	9 ± 4	1 ± 2	0 ± 0	10 ± 3
HBI	MMSeqs2 (sequence)	52 ± 7	36 ± 6	29 ± 6	35 ± 6	38 ± 6
	HMMER (profile)	70 ± 6	60 ± 6	59 ± 7	77 ± 7	67 ± 6
EAT - unsupervised	ProSE-DLM	74 ± 6	48 ± 7	28 ± 6	25 ± 7	44 ± 6
	ESM-1b	79 ± 5	61 ± 6	50 ± 7	57 ± 8	62 ± 7
	ProtBERT	67 ± 6	38 ± 6	22 ± 6	18 ± 6	36 ± 6
	ProtT5	84 ± 5	67 ± 6	57 ± 6	64 ± 8	68 ± 6
EAT - supervised	ProSE-MT	82 ± 5	65 ± 6	52 ± 7	56 ± 8	64 ± 7
EAT - contrastive learning—ProtTucker	ProSE-DLM	78 ± 4	53 ± 6	32 ± 6	29 ± 7	48 ± 6
	ProSE-MT	87 ± 4	68 ± 6	53 ± 7	55 ± 8	66 ± 6
	ESM-1b	87 ± 4	68 ± 6	59 ± 7	70 ± 7	71 ± 6
	ProtBERT	81 ± 5	52 ± 7	37 ± 6	39 ± 8	52 ± 7
	ProtT5	**89 ± 4**	75 ± 6	64 ± 6	76 ± 6	76 ± 6
	ProtT5 (train11M)	88 ± 4	**77 ± 5**	**68 ± 5**	**79 ± 7**	**78 ± 6**

*Accuracy (Equation 3) for predicting CATH (54,1) levels (C, A, T, H) by transferring annotations from *Lookup69k* (lookup set) to *test219* (queries); shown for each of the four levels from the most coarse-grained level class C to the most fine-grained level of homology H. The column *Mean* marked the simple arithmetic average over the four performance values. Queries with at least one lookup protein of the same CATH classification but without any hit at E-value < 10 for MMSeqs/HMMER were counted as incorrect predictions. Errors indicate bootstrapped 95% confidence intervals, i.e. 1.96 bootstrap standard errors (Equation 6). Queries with at least one lookup protein of the same CATH annotation but without any hit (no hit with E-value < 10 for MMSeqs/HMMER; irrelevant for EAT) were counted as wrong predictions. **Bold** letters mark the numerically highest values (averages over all *test219* proteins) in each column irrespective of the confidence interval.

Methods: Baseline: Random transferred the label of a randomly picked protein; HBI: MMSeqs2 (27) used single sequence search to transfer the annotation of the hit with the lowest *E*-value; HBI: HMMER used HMM-profiles (74); EAT-unsupervised: embedding-based transfer of annotations using the smallest Euclidean distance measured in embedding space of unsupervised pLMs ProSE-DLM, ESM-1b (38), ProtBERT and ProtT5 (37); EAT-supervised: annotation transfer using ProSE-MT trained on structural data in SCOPe; EAT: contrastive learning ProtTucker: contrastive learning trained on CATH classifications in *train66k* using as input embeddings from ProSE-DLM, ProSE-MT, ESM-1b, ProtBERT and ProtT5; ProtTucker-ProtT5 (train*11M*) trained on additional data from Gene3D (*train11M*).

EAT improved through supervised embeddings

ProSE-MT expands ProSE-DLM by additionally training on intra-residue contacts and structural similarity using labeled data from SCOPe (44). This additional effort was reflected by the higher performance for all CATH levels (Table 1, ProSE-MT > ProSE-DLM). The supervision pushed the LSTM-based ProSE-MT to reach performance levels close to the unsupervised, raw embeddings from transformer-based ProtT5. The performance gap increased with classification difficulty (Table 1, ProtT5 > ProSE-MT, especially at the H-level).

EAT improved by contrastively learning embeddings

Contrastive learning tries to bring members from the same class/CATH-level closer while pushing those from different classes further apart. One success is the degree to which these two distributions (same versus different) were separated through training: the distribution of all pairwise Euclidean distances within (intra/same) and between (inter/different) superfamilies in *train66k* changed substantially through contrastive learning (Figure 2). Before applying contrastive learning, the distributions between (inter, Figure 2: red) and within (intra, Figure 2: blue) overlapped much more than after.

Displaying the information learned by the embeddings, we compared t-SNE projections colored by the four main CATH classes before (Figure 3A) and after (Figure 3C) contrastive learning. These two projections compared 1024 dimensions from ProtT5 (Figure 3A) with 128 dimensions from ProtTucker (Figure 3C). To rule out visual effects from higher dimensionality, we also compared the untrained, randomly initialized version of ProtTucker using pre-trained ProtT5 embeddings as input (Figure 3B). For all cases, the data set (*train66k*) and the parameters for dimensionality reduction were identical. T-SNE projections of raw ProtT5 embeddings qualitatively suggested some class separation (clustering). The information underlying this separation was preserved when projecting ProtT5 embeddings through an untrained ProtTucker (Figure 3B). Embeddings from ProtTucker(ProtT5), i.e. those resulting through contrastive learning, separated the classes much more clearly.

To further probe the extent to which contrastive learning captured remote homologs, we compared the Euclidean distance between all protein pairs in a 30% non-redundant dataset (CATH-S30) with the structural similarity of those pairs computed via SSAP (65,66) (Figure 4). From the ∼10M possible pairs between the 3,186 proteins in CATH-S30 (problem not fully symmetric, therefore N*(N - 1): 10.1M), 7.1M had to be discarded due to low quality (SSAP-score < 50), leaving 2.9M pairs of which only 1.8% (53k pairs) had the same homologous superfamily (Figure 4: blue). Despite this imbalance, unsupervised ProtT5 (Figure 4A) already to some extent separated the same from different superfamilies. Still, ProtTucker(ProtT5) improved this separation, especially, for pairs with low structural similarity (Figure 4B). This was supported by the Spearman correlation coefficient between the structural similarity and the Euclidean distance increasing from 0.05 to 0.22 after contrastive learning. When

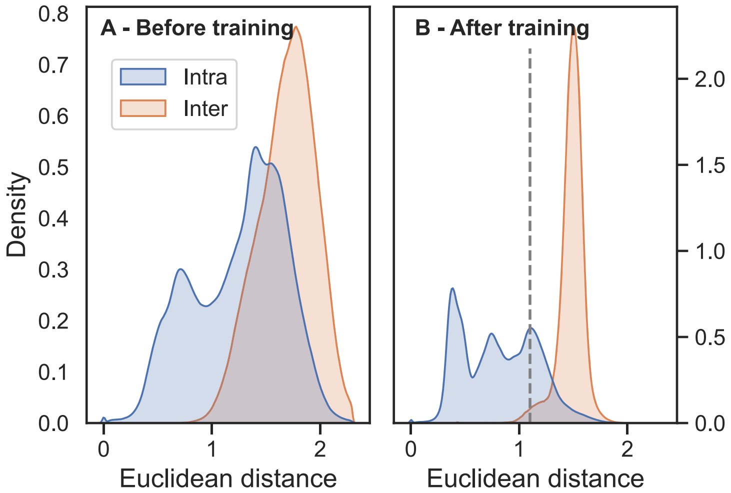

Figure 2. Contrastive learning separated positives from negatives. The structural similarity defined by increasing overlap in CATH drove the contrastive learning of a new embedding space. The new embeddings distanced protein pairs with different CATH classifications (red; same topology but different superfamily) while focusing pairs with the same CATH classification (blue; same superfamily). These graphs compared the Euclidean distance for all such pairs from the set *train66k* before (Panel **A**) and after (Panel **B**) contrastive training. Input to the FNN were the raw embeddings from ProtT5 (37), output were the new ProtTucker(ProtT5) embeddings. The dashed line at Euclidean distance of 1.1 in B marked the threshold at which EAT performances started to decrease (Figure 5).

considering only the subset of pairs that likely have similar folds (SSAP-score > 70), this correlation increased to 0.26 and 0.37 for ProtT5 and ProtTucker(ProtT5), respectively.

The trend captured by the better separation of distributions (Figure 2) and structural features (Figures 3 and 4) translated directly into performance increases: all embeddings optimized on the CATH hierarchy through contrastive learning yielded better EAT classifications than the raw embeddings from pre-trained pLMs (Table 1). As ProtTucker described the process of refining raw embeddings through contrastive learning, we used the annotation ProtTucker(*X*)—in this section also shortened to PT(X)—to refer to the embeddings output by inputting the pre-trained pLM *X* into the contrastive learning. The improvements were larger for more fine-grained CATH levels: all models improved significantly for the H-level, while only PT(ProtBERT) and PT(ESM-1b) improved from 4 to 14 or from 0 to 21 percentage points for the C-, and the H-level, respectively. PT(ProtT5) consistently outperformed all other pLMs on all four CATH-levels, with an increasing performance gap toward the more fine-grained H-level at which all pLMs except for PT(ESM-1b) performed significantly worse. The improvements from contrastive learning for PT(ProSE-DLM) and PT(ProSE-MT) were mostly consistent but largely insignificant. Especially, the model already optimized using labeled data (ProSE-MT) hardly improved through another round of supervision by contrastive learning and even worsened slightly at the H-level.

We augmented the training set for PT(ProtT5) by adding HBI-hits from HMM-profiles provided by CATH-Gene3D (if sequence dissimilar to test300/val200). This increased the training set from 66k (66 × 10^3) to 11m (11 × 10^6) proteins (15-fold increase) and raised performance, although the higher values were neither statistically significant nor consistent (Table 1: values in last row not always higher than those in second to last row).

Ablation study

We studied the effect of *batch-hard* and *hierarchical* sampling on performance by removing each component when training PT(ProtT5) (Table 2). Benchmarking on EAT from *lookup69k* to *test219* established that removing either component lowered performance for all CATH levels. Dropping both sampling methods substantially lowered performance. While dropping *batch-hard* sampling still reached high performance for the coarse-grained C- and A-level, dropping hierarchy-sampling dropped performance for both. Dropping both sampling technique, performed worse for all CATH levels but the decrease for the more fine-grained superfamily level was much larger than for the C-level.

Embedding distance correlated with accuracy

The MCC, (Equation 5) of HBI inversely correlated with *E*-value (Figure 6, HBI-methods): more significant hits (lower E-values) more often shared the same CATH level than less significant hits (higher *E*-values). In analogy, we explored the corresponding relation for EAT, namely the correlation between accuracy (Equation 3) and embedding distance for ProtTucker(ProtT5). Indeed, accuracy correlated with embedding distance (Figure 5: solid lines) while recall inversely correlated (Figure 5: dashed lines) for all four classes. For instance, when transferring only annotations for closest hits

Table 2. Ablation study*

	C	A	T	H	Mean
Baseline	89 ± 4	75 ± 6	64 ± 6	76 ± 6	76 ± 6
-batch-hard	88 ± 4	73 ± 6	62 ± 7	69 ± 7	73 ± 6
-hierarchy	83 ± 5	69 ± 6	62 ± 7	71 ± 7	71 ± 6
-both	83 ± 5	63 ± 6	51 ± 7	57 ± 8	64 ± 7

*Accuracy (Equation 3) and 95% CI (Equation 6) for predicting CATH-levels (54,1) through EAT from *Lookup69k* (lookup set) to *test219* (queries). We investigate the effect on performance when dropping either *batch-hard* sampling, *hierarchy*-sampling or *both* from the *Baseline* model (ProtTucker(ProtT5)).

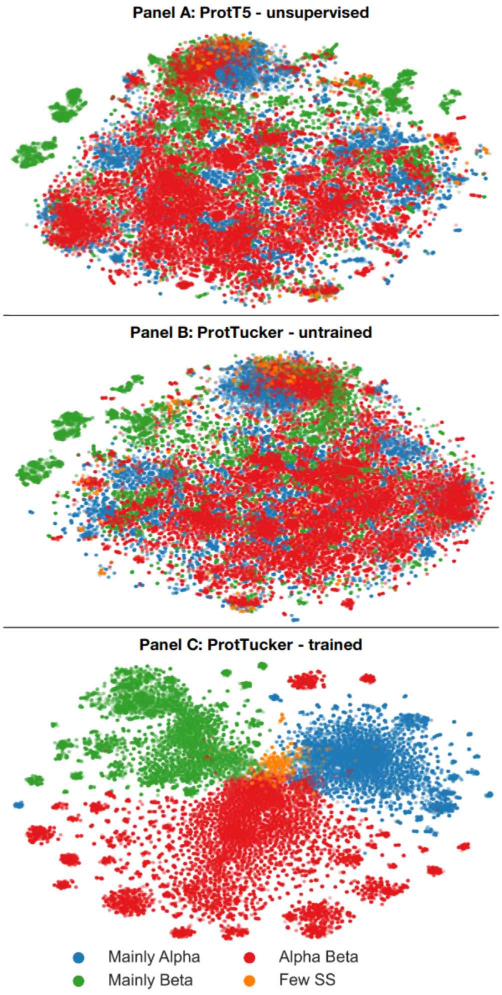

Figure 3. Better CATH class-level clustering. Using t-SNE (86), we projected the high-dimensional ProtTucker(ProtT5) embedding space onto 2D *before* (Panel **A**; ProtT5) and *after* (Panel **C**; ProtTucker(ProtT5)) contrastive learning. Panel **B** visualized the same data embedded with an untrained version of ProtTucker to assess the impact of different embedding dimensions (1024-d for ProtT5 versus 128-d for ProtTucker(ProtT5)). For all plots, dimensionality was first reduced by Principal Component Analysis (PCA) to 50 dimensions and parameters of the subsequent t-SNE were identical (perplexity = 150, learning_rate = 400, n_iter = 1000, seed = 42). The colors mark the major class level of CATH (C), distinguishing proteins according to their major distinction in secondary structure content.

with Euclidean distances of 1.1 or less, predictions were made for 57%, 57%, 59% or 75% of the test set (coverage, Equation 4) of these 96%, 93%, 91% or 90% were correct for levels C, A, T, H, respectively.

ProtTucker reached into the midnight zone

Annotation transfer by HBI crucially depends on the sequence similarity between query (unknown annotation) and template (experimental annotation). Usually, the significance of an inference is measured as the chance of finding a hit at random for a given database size (*E*-value; the lower the better). Here, we compared the effect of gradually removing hits depending on their *E*-values. Essentially, this approach measured how sensitive performance was to the degree of redundancy reduction between query and lookup set. For instance, at a value of 10^{-3} (dashed vertical lines in Figure 6), all pairs with *E*-values $\leq 10^{-3}$ were removed. HBI based on sequence alone performed much better with than without residual redundancy (Figure 6). The trend was similar for EAT, but much less pronounced: EAT succeeded for pairs with very different sequences (Figure 6 toward right) almost as well as for proteins with more sequence similar matches in the set (Figure 6 toward left: EAT almost as high as toward right).

ProtTucker not a generalist

We evaluated the generality of ProtTucker embeddings by (mis)-using them as exclusive input to predict subcellular location in ten states. To this end, we EAT transferred annotations from an established data set (Supplementary Table S5) to a strictly non-redundant test set (*setHard*, Supplementary Table S5). ProtTucker(ProtT5) embeddings outperformed the raw ProtT5 embeddings in the CATH classification for which they were optimized (structural similarity; Table 1), there appeared no performance gain in predicting location. Conversely, performance also did not decrease significantly, indicating that the new embeddings retained some of the information available in ProtT5 embeddings.

Family size mattered

By clustering very large protein families (>100 members after redundancy reduction) at 95% PIDE, we constrained the redundancy in set *train66k*. Nevertheless, when splitting *test219* into three bins of varying family sizes, we still observed a trend towards higher accuracy (Equation 3) for larger families at the H-level (Supplementary Figure S1). We chose the three bins such that they contained about

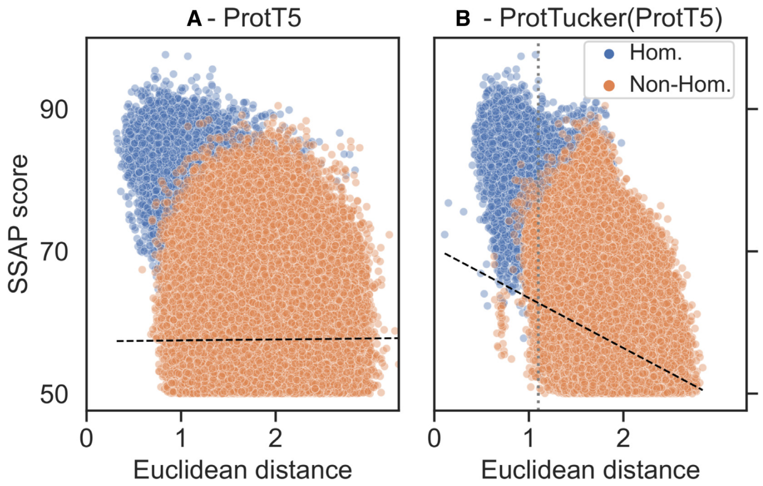

Figure 4. ProtTucker captured fine-grained structural similarity. 3186 non-redundant proteins (CATH-S30) probed the remote homology detection of embeddings *before* (Panel **A**) and *after* contrastive learning (Panel **B**). The Euclidean distance between ProtTucker embeddings (Panel B) correlated better with structural similarity computed by SSAP (65,66) than unsupervised embeddings (Panel A): Spearman ρ = 0.22 and ρ = 0.05 (black dashed lines). This correlation increased to ρ = 0.37 and ρ = 0.26 for structurally more similar protein pairs (SSAP-score > 70). Only 1.8% (53k) of all structurally similar pairs were in the same homologous superfamily (blue). The unsupervised ProtT5 already separated homologous pairs from others, but ProtTucker(ProtT5) improved, especially, for hard cases with low structural similarity. The gray dashed line at Euclidean distance = 1.1 in Panel B marked the threshold at which EAT performances started to decrease (Figure 5).

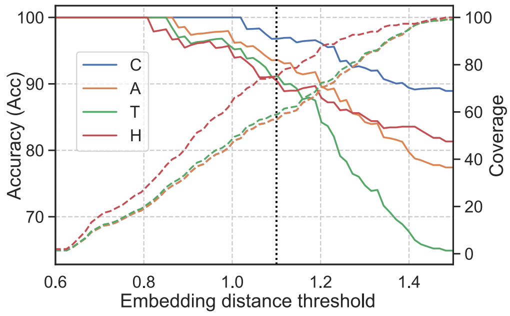

Figure 5. Embedding distance correlated with reliability. Similar to varying *E*-value cut-offs for HBI, we examined whether the fraction of correct predictions (accuracy; left axis; Equation 3) depended on embedding distance (x-axis) for EAT. This was shown by transferring annotations for all four CATH levels (Class: blue; Architecture: orange; Topology: green; Homologous superfamily: red) from *lookup69k* to the queries in set *test219* (Panel **B** in Figure 1) using the hit with lowest Euclidean distance. The fraction of *test219* proteins having a hit below a certain distance threshold (coverage, right axis, dashed lines; Equation 4) was evaluated separately for each CATH level. For example, at an Euclidean distance of 1.1 (vertical dotted line), 75% of the proteins found a hit at the *H*-level (Cov(H) = 75%) and 90% were correctly predicted (Acc(*H*) = 90%; SOM Tables S3 and S4 for more details). Thus, decreasing embedding distance correlated with EAT performance. This correlation enables users to select only the, e.g. 10% top hits, or as many hits to a certain CATH level as possible, depending on the objectives.

the same number of samples (small families: ≤10 members, medium: 11–70, and large: ≥70 members). Especially, unsupervised EAT using the raw ProtT5 embeddings exhibited a clear trend towards higher accuracy with increasing family size. In contrast, the two HBI-methods (MMseqs2, HMMER), as well as EAT using the optimized ProtTucker(ProtT5) embeddings performed similarly for small and medium-sized families and much better for large families.

EAT complements HBI

As previously shown (49), ProtTucker can improve clustering functional families (76). Here, we showed how EAT can be used to detect outliers. Firstly, we computed pairwise Euclidean distances between the embeddings of all protein pairs in set *train66k* and analyzed the five pairs (10 proteins) with the highest Euclidean distance in the same homologous superfamily (Supplementary Table S6). High distance within the same homologous superfamily indicates potentially wrong annotations. Secondly, we computed the nearest neighbors of those ten proteins to find an alternative, potentially more suitable annotation. For instance, the proteins in the *Phosphorylase Kinase* superfamily with the largest embedding distance (4pdyA01, bacterial aminoglycoside phosphotrans-ferase) to any other protein within this family (3skjF00, human *Galactose-binding domain-like* (77)) linked to different UniProt entries (*C8WS74_ALIAD* and *EPHA2_HUMAN*). In contrast, the nearest neighbor (3heiA00, human *phosphorylase kinase* (78)) of 3skjF00 linked to the same UniProt entry (*EPHA2_HUMAN* (68)) with the same enzymatic activity (EC number 2.7.10.1 (79)). Such analyses may indicate impure homologous superfamilies along with suggesting al-

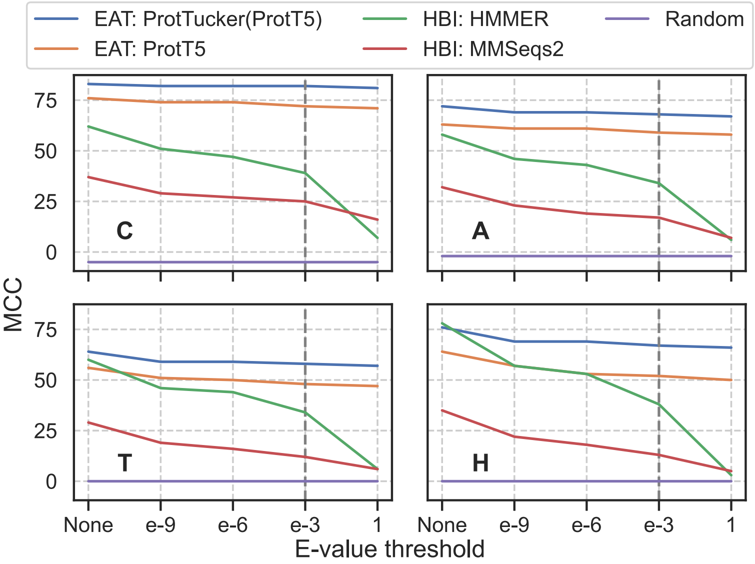

Figure 6. Performance decreasing with lower residual sequence similarity. We analyzed the change of performance in MCC (Equation 3) through removing proteins from *lookup69k* based on their *E*-value with respect to *test219* for two HBI-based (green: HMMER (74); red: MMSeqs2 (27)) and two EAT-based methods (orange: raw ProtT5 (37); blue: contrastive learning optimized ProtTucker(ProtT5)). The *E*-values were derived by searching sequences in *test219* against *lookup69k* using (i) HMM-profiles from CATH-Gene3D (54) through HMMer and (ii) MMSeqs2 sequence search with highest sensitivity (*-s 7.5, -cov 0*). 'None' referred to the performance without applying any threshold, i.e. all proteins in lookup69k were used for annotation transfer; all other thresholds referred to removing proteins below this *E*-value from *lookup69k*. Predictions were considered as false positives when no hit was found; pairs without CATH class matches were ignored. While the performance of EAT using raw ProtT5 and refined ProtTucker(ProtT5) embeddings decreased upon removing sequence similar pairs (toward right), HBI-based methods dropped significantly more. The default threshold for most sequence searches (*E*-value < 1e–3) was highlighted by vertical, gray, dashed lines.

ternative labels to be confirmed or rejected through manual curation.

EAT predicts entire proteomes in minutes

Training ProtTucker(ProtT5) required generating ProtT5 embeddings for *train66k*. This took 23m and 11m, respectively. Embeddings were generated using ProtT5 in half-precision with batch processing. All times were measured on a single Nvidia RTX A6000 with 48GB of vRAM and an AMD EPYC ROME 7352.

When predicting for new queries, ProtTucker requires *labeled lookup* proteins from which annotations can be transferred to unlabeled query proteins. Embeddings for this lookup set are pre-computed for the first query and can be re-used for all subsequent queries at any future time. The time required to labeled *lookup* proteins from which annotations can be transferred to unlabeled query proteins. Embeddings for this lookup set are pre-computed for the first query and can be re-used for all subsequent queries at any future time. The time required to generate ProtTucker embeddings from the embeddings of pLMs was negligible as its generation required only a single forward pass through a two-layer FNN. This implied that the total time for EAT with ProtTucker was largely determined by the embedding generation speed. For instance, creating per-protein embeddings from ProtT5 for the 123k proteins in CATH-S100 required 23 min (m). The total time for creating ProtTucker(ProtT5) embeddings from ProtT5 embeddings for the same set on the same machine was 0.5 seconds (s), i.e. ProtTucker added about 0.3%. Creating HMM profiles for the same set using either MSAs from MMSeqs2 (–*num-iterations 3, -s 7.5*) or jackhmmer took 15 m or 30 h, respectively (Table 3).

Table 3. Runtime*

Methods	Pre-processing (s)	Inference (s)
MMSeqs2 (sequence)	0.2×10^3	2.5×10^{-2}
HMMER (HMM)	114×10^3	150×10^{-2}
ProtTucker(ProtT5)	1.4×10^3	1.4×10^{-2}

* Runtime to transfer annotations from CATH-S100 (123k proteins) to a single query. All times measured in seconds [s] on a single Nvidia RTX A6000 with 48GB of vRAM and an AMD EPYC ROME. *Methods*: two HBI-based methods (MMSeqs2 and HMMER) and one EAT-based method (ProtTucker(ProtT5)). Pre-processing: measured the time required for building datasets (indexed database for MMSeqs2; MSA for jackhammer plus HMM profiles (HMMER) or ProtT5 embeddings (ProtTucker); Inference: the time for each new protein with a transfer.

To predict using EAT, users have to embed only single query proteins requiring, on average, 0.01 s per protein for the CATH-S100 set. Using either single protein sequence search (MMSeqs2), pre-computed HMM profiles (HMMER) or pre-computed embeddings (ProtTucker) to transfer annotations from CATH-S100 to a single query protein took on average 0.025, 1.5 or 0.0008 s, respectively. Proteins in the PDB and CATH are, on average, roughly half as long (173 residues) as those from UniProt (343 residues). This is relevant for runtime, because embedding generation scales quadratically with sequence length (Figure 13 in SOM of (37)).

This increase was also reflected for the proteome-wide annotation transfer (Table 4), although these values included computations required for all aspects of EAT (1: load ProtT5 embeddings for pre-computed CATH-S100 lookup set; 2: load ProtT5 and embedding for query proteome; 3: generate ProtTucker(ProtT5) embeddings for queries and lookup; 4: compute pairwise Euclidean distances between query/lookup). We compared EAT using ProtTucker(ProtT5) embeddings to HBI proxied by existing Gene3D annotations taken from UniProt for three different proteomes (Table 4). At an expected error rate of 5% (Euclidean distance $\leq$ 0.9, Supplementary Table S3), EAT predicted substantially more proteins than Gene3D at HMMER *E*-value $< 10^{-3}$. For the subset of proteins for which both methods transferred annotations, those largely agreed (*Agreement*, Table 4; Supplementary Table S7 for other thresholds). All values for coverage decreased for multi-domain proteins, as proxied by ‘multiple Gene3D annotations’, while the agreement between Gene3D and ProtTucker(ProtT5) increased for two of three proteomes (Table 4: *multi*).

DISCUSSION

Prototype for representation learning of hierarchies

We have presented a new solution for combining the information implicitly contained in the embeddings from protein Language Models (pLMs) and contrastive learning to learn directly from hierarchically sorted data. As proof-of-concept, we applied the concept to the CATH hierarchy of protein structures (54,1,6,80). Hierarchies are difficult to handle by traditional supervised learning solutions. One *shortcut* is to learn each level in the hierarchy independently (81–83) at the price of having less information for other levels and of not explicitly benefiting from the hierarchy. Instead, our solution of contrastively learning protein triplets (anchor, positive, negative) to extract a new embedding space by condensing positives and moving negatives apart benefits from CATH’s hierarchical structure. Simultaneously training a single, shared feed-forward neural network (FNN) on triplets from all four CATH classification levels allowed the network to directly capture the hierarchy. Encoding protein sequences through previously trained pLMs enabled ready information transfer from large but unlabeled sequence databases such as BFD (69) to 10,000-times smaller but experimentally annotated (labeled) proteins of known 3D structure classified by CATH. In turn, this allowed us to readily leverage aspects of protein structure captured by pLMs that are informative enough to predict structure from embeddings alone (52). Although the raw, pre-trained, unoptimized embeddings captured some aspects of the classification (Figures 2A–4A, Table 1), contrastive learning boosted this signal significantly (Figures 2B–4B, Table 1).

Crucial for this success was the novel combination of hierarchy- and batch-hard sampling (Table 2). Presumably, because those techniques enforce so-called *semi-hard* triplets that are neither too simple nor too hard to learn (72). This training setup learned different classifications for the same protein pair, depending on the third protein forming the triplet, thereby forcing the network to learn the complex hierarchy. The ambivalence in the notion of *positive/negative pair* facilitated training by allowing to include superfamilies with few members (otherwise to be skipped) and it increased the number of possible triplets manifold compared to only sampling on the level of superfamilies. These advantages might partially explain the synergy of both sampling techniques (Table 2).

Raw embedding EAT matched profile alignments in hit detection

In technical analogy to homology-based inference (HBI), we used embedding based annotation transfer (EAT, Figure 1B) to transfer annotations from labeled lookup proteins (proteins with a known CATH classification) to unlabeled query proteins (any protein of known sequence without known structure). Instead of transferring annotations from the closest hit in sequence space, EAT transferred annotations to the hit with smallest Euclidean distance in embedding space. This relatively simple approach was shown previously to predict protein function as defined by Gene Ontology (GO) better than hand-crafted features (50) even to levels competitive to much more complex approaches (48).

The concept of EAT was so successful that raw embeddings from two different pre-trained pLMs (ESM-1b (38), and ProtT5 (37)) already set the bar high for predicting CATH levels. The raw, general-purpose ESM-1b and ProtT5 outperformed HBI based on advanced HMM-profiles from HMMER (74) on the C- and A-level while falling short on the H-level (Table 1). Furthermore, we showed that ProtT5 already separated protein pairs with the same from those with different homologous superfamilies even when using a lookup set that consisted only of proteins with maximally 30% pairwise sequence identity (Figure 4A). Importantly, this competitive performance was achieved at a much smaller cost in terms of runtime (Tables 3 and 4).

As the lookup embeddings or HMM profiles are computed only once, we neglected this additional step. Such preparations cost much more than single queries: pre-computing HMM profiles using MMseqs2 took 15m, pre-computing embeddings about 23m (Table 3) using the same set and machine but utilizing CPUs in one (MMseqs2) and GPUs in the other (ProtT5). Only MMSeqs2 generated and indexed its database rapidly (19.5 s). However, pre-processing is required only once, rapidly amortizing when running many queries. The ability to pre-compute such representations is also a crucial difference between ProtTucker and other learned methods (44,59). For pairwise protein comparisons, those methods typically require N comparisons/forward-passes to search with a single query against N proteins. Instead, ProtTucker only needs a single forward pass to embed the new query; subsequent similarity scoring simply and quickly computes an Euclidean distance.

Table 4. CATH predictions for three model proteomes*

Proteome	Size	Gene3D	ProtTucker(ProtT5)@0.9	Agreement	Gene3D-multi	Agreement-multi	Inference time [s]
E. Coli (K12)	2033	59% (1193)	97% (1982)	81% (963)	23% (275)	86% (235)	113 (0.06)
A. Ostoyae	22 192	31% (6902)	79% (17 416)	75% (4707)	18% (1223)	65% (684)	1384 (0.06)
M. Chiliensis	1120	35% (392)	87% (974)	84% (320)	10% (40)	73% (29)	95 (0.09)

* Comparison of the annotation-transfer from 123k CATH-S100 proteins (5,54) through HBI (*Gene3D* (64)) and through EAT as introduced here (ProtTucker(ProtT5), or PT(ProtT5)) for three entire reference proteomes: *Escherichia coli* (*E. coli*), *Armillaria ostoyae* (*A. ostoyae*) and *Megavirus chilensis* (*M. chilensis*). In other words, all proteins in the three organisms were mapped to proteins of known structure using the CATH hierarchy. Gene3D predictions were taken from UniProt; PT(ProtT5) predictions were derived from the single nearest neighbor in Euclidean space. Coverage-related numbers refer to the percentage of proteins in the entire proteome (Size; in brackets: actual number of proteins), while those pertaining to the agreement are percentages of the set with annotations. *Size:* number of proteins; *Gene3D:* fraction of proteins with Gene3D annotation (coverage); *PT(ProT5)@0.9*: coverage of PT(ProtT5) at Euclidean distance < = 0.9 (5% error rate; Supplementary Table S3); *Agreement:* fraction of proteins for which Gene3D and PT(ProT5) had a prediction and reported the same homologous CATH superfamily (for multi-domain proteins with multiple Gene3D annotations, matching any domain by PT(ProtT5) was considered as correct); *Gene3D Multi:* fraction of Gene3D proteins with multi-domain annotation; *Agreement Multi:* fraction of multi-domain proteins for which the homologous CATH superfamily predicted by PT5 agreed with one of the Gene3D domain annotations; *Inference time:* the total time needed for proteome-wide embedding-based annotation transfer (EAT) measured in seconds [s] on a single Nvidia RTX A6000 with 48GB of vRAM and an AMD EPYC ROME (in brackets the average time per protein).

This makes ProtTucker search speed scale well with database growth suggesting the tool as a fast but sensitive pre-filter for other methods that in turn provide residue-level information as showcased on three model organisms (Table 4), including one of the largest organisms on earth (fungus *A. ostoyae*, 22 192 proteins) and one of the largest viruses (*M. chilensis*, 1120 proteins). In <27 min on a single machine (Table 4), ProtTucker transferred substantially more CATH annotations mapping proteins from their sequence to 3D structures through the CATH resource than Gene3D (64) at a similar level of expected error (Table 4).

For the virus and the bacterium (*E. coli*) the annotations agreed to over 80% with Gene3D, while this value dropped to 75% for the fungus (Table 4). Although high, the agreement was lower than expected: if ProtTucker and Gene3D each had fewer than 5% errors, then both should agree for over 90% of the proteins for which both transfer annotations. Most likely, this discrepancy (Δ(90-77)) arose partially from multi-domain proteins. Despite carefully cross-validating ProtTucker, an alternative explanation for the discrepancy is underestima-ting the expected error a distances ≤0.9 as 5% instead of up to 15%. The 'functional shape' of the agreement between ProtTucker and Gene3D at different distance thresholds (Supplementary Table S7) suggested that the 'errors' (lack of agreement) did not only originate from ProtTucker. Carefully annotating the five proteins with the lowest distance and a different CATH annotation (Supplementary Table S4) supported this perspective.

The agreement for multi-domain proteins dropped less than expected (11 percentage points drop for *M. Chilensis*, 5 percentage points increase for *E. coli*), possibly suggesting that ProtTucker using averages over an entire protein for comparison did not trip substantially more over the multi-domain challenge than the local alignment-based Gene3D using HMMER (74). This might suggest ProtTucker to have added correct annotations over Gene3D in multi-domain proteins, although developed exclusively on single domain proteins. The substantial increase in coverage from the level expected at distances ≥0.9 (Figure 5, Supplementary Table S4) for the proteomes (Table 4) might be misleading: to establish performance coverage (Figure 5, Supplementary Table S4), we used a highly non-redundant lookup set, presumably removing many easy hits. In contrast, analyzing proteomes, we transferred annotations for all CATH-S100 proteins, leveraging 'redundant annotation transfers' to increase coverage.

As for HBI, the accuracy of EAT also increased for larger families (Supplementary Figure S1). One explanation is that the larger the family, the higher the random hit rate, simply because there are more possible hits. Another, more subtle (and given the enormous compute time needed to train ProtT5, more difficult to test) explanation is that the largest CATH families represent most of the largest protein families (54). In fact, a few hundred of the largest superfamilies cover half of the entire sequence space (54,84). Simply due to their immense size, these large families have been sampled more during the pre-training of ProtT5.

ProtTucker embeddings intruded into midnight zone

The embedding space resulting from contrastive learning, introduced here, improved performance consistently for all four pLMs (Table 1). This was revealed through several ways of looking at the results from embeddings with and without contrastive learning: (i) the increased separation of protein pairs within the same protein superfamily and between different superfamilies (Figure 2), (ii) the qualitative improvement in the clustering of t-SNE projections (Figure 3), the better correlation of embedding distance and structural similarity (Figure 4) and (iii) the quantitative improvement in the EAT benchmark (Table 1). On top, the Euclidean distance correlated with accuracy (Figure 5, Supplementary Table S3). Similar to an E-value in HBI, this lets users gauge the reliability of a hit between query and annotated protein.

While the accuracy of the best performing pLM (ProtTucker(ProtT5)) was similar to HBI using HMM-profiles on the most fine-grained level of homologous superfamilies (CATH level H, Table 1), the relative advantage of EAT increased, the more diverged the level of inference, i.e. EAT outperformed HBI for more distant relations from the midnight zone (CATH level C, Table 1). When further reducing data redundancy, i.e. removing more similar sequences, this trend became clearer (Figure 6). Despite increasing difficulty, the performance of EAT decreased almost insignificantly where HBI approached random for insignificant E-values. This trend was supported by the correlation of structural similarity as defined by SSAP (65,66) and the Euclidean distance between protein pairs in a 30% non-redundant data set (Figure 4).

ProtTucker and tools such as HMMer have very different resolution: ProtTucker considers only per-protein averages to match query to template. In contrast, HMMer – or similar methods – align each residue between both proteins. The coarse-grain yields the speedup (Table 4), and pitches ProtTucker as a fast pre-filter. Once the hit is found by scanning large data sets, the slower, fine-grained methods for per-residue alignments and 3D prediction can be employed. However, the per-protein average also implies limitations, e.g. when Q and T have very different numbers of domains or the number of domains for Q is not known (Table 3).

Ultimately, the coarse-grained ProtTucker can compete at all because embeddings intrinsically abstract the constraints under which protein sequences evolve, including constraints upon structure, function, and the environment. The same constraints coin the evolutionary information contained in profiles of protein families. Apparently, pLMs such as ESM-1b (38), ProtBERT (70), or ProtT5 (70) are successfully condense these constraints. In fact, pLMs are arguably more successful than profile-based methods because a simple length-average over the position-specific scoring metrices (PSSM) would not suffice to predict CATH numbers very accurately.

ProtTucker builds upon this success to explicitly capture the constraints relevant for the CATH hierarchy. Thus, the less a particular aspect of function depends on structure, the less likely the new ProtTucker embeddings will reflect this aspect. On the other hand, an approach similar to ProtTucker focused on particular functional hierarchies, e.g. EC numbers appears to work well (SM Akmese & M Heinzinger, unpublished).

Taken together, these results indicated that contrastive learning captured structural hierarchies and provides a novel, powerful tool to uncover structural similarities clearly beyond what has been achievable with 50 years of optimizing sequence-based alignment techniques. Using EAT to complement HBI could become crucial for a variety of applications, ranging from finding remote structural templates for protein 3D structure predictions over prioritizing new proteins without any similarity to an existing structure to filtering potentially wrong annotations. One particular example has recently been shown for the proteome of SARS-CoV-2 to unravel entire functional components possibly relevant for fighting COVID-19 (61).

ProtTucker embeddings improved FunFams clustering

Previously (49), we showed that a simplistic predecessor of ProtTucker helped to refine the clustering of FunFams (76). By adding an additional, more fine-grained hierarchy level in CATH, FunFams link the structure-function continuum of proteins. The functional consistency within FunFams was proxied through the enzymatic activity as defined by the EC (Enzyme Commission (79)) number. Even the preliminary ProtTucker improved the annotation transfer of ligand binding and EC numbers (49) by removing outliers from existing FunFams and by creating new, more functionally coherent FunFams. As for CATH, the contrastively trained ProtTucker(ProtBERT) also improved over its unsupervised counterpart, ProtBERT, for FunFams. It improved functional consistency especially for proteins in the twilight zone (<35% PIDE, Figure 5 in (49)). Thus, ProtTucker embeddings improved functional (FunFams) and structural (CATH) consistency beyond sequence similarity. Here, we expanded upon this analysis by showing how EAT can be improved even more through contrastively learning hierarchies. Using the proposed method, we could spot potential outliers, i.e. samples with the same annotation but large embedding distance. This might become essential to clean up databases. Aside from outlier-spotting, we could also obtain labels from the nearest neighbors of outliers (Supplementary Table S6). Although we could not reproduce the same level of success when applying EAT to inferring subcellular location in ten states (Table 3), the CATH-optimized ProtTucker embeddings also did not perform worse.

Generic advantages of *contrastive learning*

Contrastive learning benefits from hierarchies as opposed to supervised training which usually flattens the hierarchy thereby losing its intrinsic advantage. Other possible advantages of contrastive learning include the following three. (i) Dynamic data update (*online learning*): While supervised networks require re-training to benefit from new data, contrastively trained networks can benefit from new data by simply updating the lookup set. This could even add completely new classes, such as proteins for which the classification will become available only in the future. HBI shares this advantage that originates from the difference between classifying proteins into existing families versus classifying by identifying the most similar proteins in that family. (ii) Learn the access, not the data: Instead of forcing the supervised network to memorize the training data, contrastive learning teaches how to access the data stored in an external lookup set. (iii) Compression: As many other learning techniques, contrastive learning can act as a compression technique. For instance, we reduced the disk space required to store protein embeddings threefold by projecting 1024-dimensional vectors onto 128 dimensions while improving performance (Table 1). This renders new queries (inference) more efficient and enables scaling up to very large lookup sets. (iv) Interpretability: Knowing from which protein an annotation was transferred might help users benefit more from a certain prediction than just the prediction itself. For instance, knowing that an unnamed query protein shares all CATH levels with a particular glucocorticoid receptor might suggest some functional implications helping to design future experiments.

CONCLUSIONS

Embeddings from protein Language Models (pLMs) extract the information learned by these models from unlabeled protein sequences. Embedding-based Annotation

Transfer (EAT) replacing the proximity in sequence space used by homolog-based inference (HBI) through proximity in embedding space already reaches traditional alignment methods in transferring CATH annotations from a template protein with experimental annotations to an unlabeled query protein. Although not quite reaching the performance of advanced profile-profile searches by HMMer for all four CATH levels, the best embeddings surpassed HMMer for two of the four levels (C and A). When optimizing embeddings through contrastive learning for the goal of transferring CATH annotations, EAT using these new embeddings consistently outperformed all sequence comparison techniques tested. This higher performance was reached at a fraction (three orders of magnitude) of the computational time. Although the new embeddings optimized through contrastive learning for CATH did not improve performance for a completely different task, namely the prediction of subcellular location in ten classes, the CATH-optimized solution did also not perform significantly worse. Remarkably, just like HBI, the performance of EAT using the optimized ProtTucker embeddings was proportional to family size with increased accuracy for larger families.

DATA AVAILABILITY

Building on top of bio_embeddings package (85) we have made a script available that simplifies EAT https://github.com/Rostlab/EAT.

SUPPLEMENTARY DATA

Supplementary Data are available at NARGAB Online.

ACKNOWLEDGEMENTS

Thanks primarily to Tim Karl for invaluable help with hardware and software; to Inga Weise (TUM) for support with many other aspects of this work. Many thanks for both anonymous reviewers, deep kudos for the detailed help from No. 1! Last, but not least, thanks to all those who maintain public databases in particular Steven Burley (PDB, Rutgers), Alan Bridge (Swiss-Prot, SIB, Lausanne), Alex Bateman (UniProt, EBI Hinxton) and their crews, and to all experimentalists who enabled this analysis by making their data publicly available.

FUNDING

Bavarian Ministry of Education through funding to the TUM and by a grant from the Alexander von Humboldt foundation through the German Ministry for Research and Education (BMBF: Bundesministerium für Bildung und Forschung); BMBF [031L0168 and program 'Software Campus 2.0 (TUM) 2.0' 01IS17049];Deutsche Forschungsgemeinschaft [DFG-GZ: RO1320/4-1].

Conflict of interest statement. None declared.

REFERENCES

1. Das,S., Sillitoe,I., Lee,D., Lees,J.G., Dawson,N.L., Ward,J. and Orengo,C.A. (2015) CATH funfhmmer web server: protein functional annotations using functional family assignments. *Nucleic Acids Res.*, **43**, W148–W153.
2. Sonnhammer,E.L. and Kahn,D. (1994) Modular arrangement of proteins as inferred from analysis of homology. *Protein Sci.*, **3**, 482–492.
3. Bateman,A., Birney,E., Durbin,R., Eddy,S.R., Finn,R.D. and Sonnhammer,E.L. (1999) Pfam 3.1: 1313 multiple alignments and profile HMMs match the majority of proteins. *Nucleic Acids Res.*, **27**, 260–262.
4. Gough,J. and Chothia,C. (2002) SUPERFAMILY: HMMs representing all proteins of known structure. SCOP sequence searches, alignments and genome assignments. *Nucleic Acids Res.*, **30**, 268–272.
5. Orengo,C.A., Flores,T.P., Taylor,W.R. and Thornton,J.M. (1993) Identification and classification of protein fold families. *Protein Eng.*, **6**, 485–500.
6. Orengo,C.A., Michie,A.D., Jones,S., Jones,D.T., Swindells,M.B. and Thornton,J.M. (1997) CATH - a hierarchic classification of protein domain structures. *Structures*, **5**, 1093–1108.
7. Todd,A.E., Orengo,C.A. and Thornton,J.M. (2001) Evolution of function in protein superfamilies, from a structural perspective. *J. Mol. Biol.*, **307**, 1113–1143.
8. Yona,G. and Levitt,M. (2002) Within the twilight zone: a sensitive profile-profile comparison tool based on information theory. *J. Mol. Biol.*, **315**, 1257–1275.
9. Doolittle,R.F., Feng,D.-F., Johnson,M.S. and McClure,M.A. (1986) Origins and evolutionary relationships of retroviruses. *Q. Rev. Biol.*, **64**, 1–30.
10. Rost,B. (1999) Twilight zone of protein sequence alignments. *Protein. Eng.*, **12**, 85–94.
11. Rost,B. (1997) Protein structures sustain evolutionary drift. *Fold. Des.*, **2**, S19–S24.
12. Mika,S. and Rost,B. (2003) UniqueProt: creating representative protein sequence sets. *Nucleic Acids Res.*, **31**, 3789–3791.
13. Rost,B. (2002) Enzyme function less conserved than anticipated. *J. Mol. Biol.*, **318**, 595–608.
14. Nehrt,N.L., Clark,W.T., Radivojac,P. and Hahn,M.W. (2011) Testing the ortholog conjecture with comparative functional genomic data from mammals. *PLoS Comput. Biol.*, **7**, e1002073.
15. Sander,C. and Schneider,R. (1991) Database of homology-derived structures and the structural meaning of sequence alignment. *Proteins*, **9**, 56–68.
16. Higgins,D.G., Bleasby,A.J. and Fuchs,R. (1992) CLUSTAL V: improved sofware for multiple sequence alignment. *CABIOS*, **8**, 189–191.
17. Thompson,J., Higgins,D. and Gibson,T. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4690.
18. Sjölander,K., Karplus,K., Brown,M.P., Hughey,R., Krogh,A., Mian,I.S. and Haussler,D. (1996) Dirichlet mixtures: a method for improving detection of weak but significant protein sequence homology. *CABIOS*, **12**, 327–345.
19. Altschul,S.F., Madden,T.L., Schaeffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D.J. (1997) Gapped blast and PSI-Blast: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
20. Eddy,S.R. (1998) Profile hidden markov models. *Bioinformatics*, **14**, 755–763.
21. Jaroszewski,L., Rychlewski,L. and Godzik,A. (2000) Improving the quality of twilight-zone alignments. *Protein Sci.*, **9**, 1487–1496.
22. Sadreyev,R. and Grishin,N. (2003) COMPASS: a tool for comparison of multiple protein alignments with assessment of statistical significance. *J. Mol. Biol.*, **326**, 317–336.
23. Edgar,R.C. and Sjolander,K. (2004) COACH: profile-profile alignment of protein families using hidden markov models. *Bioinformatics*, **20**, 1309–1318.
24. Wang,G. and Dunbrack,R.L. Jr (2004) Scoring profile-to-profile sequence alignments. *Protein Sci.*, **13**, 1612–1626.
25. Soding,J. (2005) Protein homology detection by HMM-HMM comparison. *Bioinformatics*, **21**, 951–960.
26. Sievers,F., Wilm,A., Dineen,D., Gibson,T.J., Karplus,K., Li,W., Lopez,R., McWilliam,H., Remmert,M., Soding,J. *et al.* (2011) Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Mol. Syst. Biol.*, **7**, 539.

27. Steinegger,M. and Soding,J. (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, **35**, 1026–1028.
28. Przybylski,D. and Rost,B. (2007) Consensus sequences improve PSI-BLAST through mimicking profile-profile alignments. *Nucleic Acids Res.*, **35**, 2238–2246.
29. Rost,B., Liu,J., Nair,R., Wrzeszczynski,K.O. and Ofran,Y. (2003) Automatic prediction of protein function. *Cell. Mol. Life Sci.*, **60**, 2637–2650.
30. Rost,B. (1996) PHD: predicting one-dimensional protein structure by profile based neural networks. *Meth Enzymol*, **266**, 525–539.
31. Rost,B. and Sander,C. (1993) Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.*, **232**, 584–599.
32. Jumper,J., Evans,R., Pritzel,A., Green,T., Figurnov,M., Ronneberger,O., Tunyasuvunakool,K., Bates,R., Zidek,A., Potapenko,A. *et al.* (2021) Highly accurate protein structure prediction with alphafold. *Nature*, **569**, 583–589.
33. Baek,M., Dimaio,F., Anishchenko,I., Dauparas,J., Ovchinnikov,S., Lee,G.R., Wang,J., Cong,Q., Kinch,L.N., Schaeffer,R.D. *et al.* (2021) Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, **373**, 871–876.
34. Peters,M.E., Neumann,M., Iyyer,M., Gardner,M., Clark,C., Lee,K. and Zettlemoyer,L. (2018) Deep contextualized word representations. arXiv doi: https://doi.org/10.48550/arXiv.1802.05365, 22 March 2018, preprint: not peer reviewed.
35. Devlin,J., Chang,M.-W., Lee,K. and Toutanova,K. (2019) *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, pp. 4171–4186.
36. Brown,T.B., Mann,B., Ryder,N., Subbiah,M., Kaplan,J., Dhariwal,P., Neelakantan,A., Shyam,P., Sastry,G., Askell,A. *et al.* (2020) Language models are few-shot learners. arXiv doi: https://doi.org/10.48550/arXiv.2005.14165, 22 July 2020, preprint: not peer reviewed.
37. Elnaggar,A., Heinzinger,M., Dallago,C., Rehawi,G., Wang,Y., Jones,L., Gibbs,T., Feher,T., Angerer,C., Steinegger,M. *et al.* (2021) ProtTrans: towards cracking the language of life's code through self-supervised learning. *IEEE TPAMI*, **14**, 30.
38. Rives,A., Meier,J., Sercu,T., Goyal,S., Lin,Z., Liu,J., Guo,D., Ott,M., Zitnick,C.L., Ma,J. *et al.* (2021) Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl Acad. Sci. U.S.A.*, **118**, e2016239118.
39. Alley,E.C., Khimulya,G., Biswas,S., AlQuraishi,M. and Church,G.M. (2019) Unified rational protein engineering with sequence-based deep representation learning. *Nature Meth*, **16**, 1315–1322.
40. Heinzinger,M., Elnaggar,A., Wang,Y., Dallago,C., Nechaev,D., Matthes,F. and Rost,B. (2019) Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinf.*, **20**, 723.
41. Rao,R., Meier,J., Sercu,T., Ovchinnikov,S. and Rives,A. (2020) Transformer protein language models are unsupervised structure learners. bioRxiv doi: https://doi.org/10.1101/2020.12.15.422761, 15 December 2020, preprint: not peer reviewed.
42. Madani,A., McCann,B., Naik,N., Shirish Keskar,N., Anand,N., Eguchi,R.R., Huang,P. and Socher,R. (2020) ProGen: language modeling for protein generation. arXiv doi: https://doi.org/10.48550/arXiv.2004.03497, 8 March 2020, preprint: not peer reviewed.
43. Ofer,D., Brandes,N. and Linial,M. (2021) The language of proteins: NLP, machine learning & protein sequences. *Comp Structural Biotechn J*, **19**, 1750–1758.
44. Bepler,T. and Berger,B. (2021) Learning the protein language: evolution, structure, and function. *Cell Syst.*, **12**, 654–669.
45. Bepler,T. and Berger,B. (2019) Learning protein sequence embeddings using information from structure. *Seventh International Conference on Learning Representations*.
46. Stärk,H., Dallago,C., Heinzinger,M. and Rost,B. (2021) Light attention predicts protein location from the language of life. *Bioinformatics Adv.*, **1**, vbab035..
47. Littmann,M., Heinzinger,M., Dallago,C., Weissenow,K. and Rost,B. (2021) Protein embeddings and deep learning predict binding residues for various ligand classes. *Sci. Rep.*, **11**, 23916.
48. Littmann,M., Heinzinger,M., Dallago,C., Olenyi,T. and Rost,B. (2021) Embeddings from deep learning transfer GO annotations beyond homology. *Sci. Rep.*, **11**, 1160.
49. Littmann,M., Bordin,N., Heinzinger,M., Schütze,K., Dallago,C., Orengo,C. and Rost,B. (2021) Clustering funfams using sequence embeddings improves EC purity. *Bioinformatics*, **37**, 3449–3455.
50. Villegas-Morcillo,A., Makrodimitris,S., van Ham,R.C.H.J., Gomez,A.M., Sanchez,V. and Reinders,M.J.T. (2021) Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, **37**, 162–170.
51. Hamid,M.-N. and Friedberg,I. (2019) Identifying antimicrobial peptides using word embedding with deep recurrent neural networks. *Bioinformatics*, **35**, 2009–2016.
52. Weißenow,K., Heinzinger,M. and Rost,B. (2022) Protein language model embeddings for fast, accurate, alignment-free protein structure prediction. *Structure*, https://doi.org/10.1016/j.str.2022.05.001.
53. Le-Khac,P.H., Healy,G. and Smeaton,A.F. (2020) *Contrastive Representation Learning: A Framework and Review*. IEEE Access.
54. Sillitoe,I., Bordin,N., Dawson,N., Waman,V.P., Ashford,P., Scholes,H.M., Pang,C.S.M., Woodridge,L., Rauer,C., Sen,N. *et al.* (2021) CATH: increased structural coverage of functional space. *Nucleic Acids Res.*, **49**, D266–D273.
55. Fox,N.K., Brenner,S.E. and Chandonia,J.-M. (2014) SCOPe: structural classification of Proteins—extended, integrating SCOP and ASTRAL data and classification of new structures. *Nucleic Acids Res.*, **42**, D304–D309.
56. Nallapareddy,V., Bordin,N., Sillitoe,I., Heinzinger,M., Littmann,M., Waman,V., Sen,N., Rost,B. and Orengo,C. (2022) CATHe: detection of remote homologues for CATH superfamilies using embeddings from protein language models. bioRxiv doi: https://doi.org/10.1101/2022.03.10.483805,13 March 2022, preprint: not peer reviewed.
57. Li,C.-C. and Liu,B. (2019) MotifCNN-fold: protein fold recognition based on fold-specific features extracted by motif-based convolutional neural networks. *Brief Bioinform*, **21**, 2133–2141.
58. Liu,B., Li,C.-C. and Yan,K. (2020) DeepSVM-fold: protein fold recognition by combining support vector machines and pairwise sequence similarity scores generated by deep learning networks. *Brief Bioinform*, **21**, 1733–1741.
59. Gao,M. and Skolnick,J. (2021) A novel sequence alignment algorithm based on deep learning of the protein folding code. *Bioinformatics*, **37**, 490–496.
60. Chen,J., Guo,M., Wang,X. and Liu,B. (2018) A comprehensive review and comparison of different computational methods for protein remote homology detection. *Brief Bioinform*, **19**, 231–244.
61. O'Donoghue,S.I., Schafferhans,A., Sikta,N., Stolte,C., Kaur,S., Ho,B.K., Anderson,S., Procter,J., Dallago,C., Bordin,N. *et al.* (2021) SARS-CoV-2 structural coverage map reveals viral protein assembly, mimicry, and hijacking mechanisms. *Mol. Syst. Biol.*, **12**, e10079.
62. Burley,S.K., Berman,H.M., Bhikadiya,C., Bi,C., Chen,L., Di Costanzo,L., Christie,C., Dalenberg,K., Duarte,J.M., Dutta,S. *et al.* (2019) RCSB protein data bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic Acids Res.*, **47**, D464–D474.
63. Chen,T., Kornblith,S., Norouzi,M. and Hinton,G. (2020) *International Conference on Machine Learning*. PMLR, pp. 1597–1607.
64. Lewis,T.E., Sillitoe,I., Dawson,N., Lam,S.D., Clarke,T., Lee,D., Orengo,C. and Lees,J. (2018) Gene3D: extensive prediction of globular domains in proteins. *Nucleic Acids Res.*, **46**, D435–D439.
65. Taylor,W.R. and Orengo,C.A. (1989) A holistic approach to protein structure alignment. *Protein. Eng.*, **2**, 505–519.
66. Orengo,C.A. and Taylor,W.R. (1996) SSAP: sequential structure alignment program for protein structure comparison. *Meth Enzymol*, **266**, 617–635.
67. Almagro Armenteros,J.J., Sonderby,C.K., Sonderby,S.K., Nielsen,H. and Winther,O. (2017) DeepLoc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, **33**, 3387–3395.
68. The UniProt Consortium. (2021) UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.*, **49**, D480–D489.

69. Steinegger,M., Mirdita,M. and Soding,J. (2019) Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods*, **16**, 603–606.
70. Raffel,C., Shazeer,N., Roberts,A., Lee,K., Narang,S., Matena,M., Zhou,Y., Li,W. and Liu,P.J. (2020) Exploring the limits of transfer learning with a unified Text-to-Text transformer. *J Mach Learning Res*, **21**, 1–67.
71. Marquet,C., Heinzinger,M., Olenyi,T., Dallago,C., Erckert,K., Bernhofer,M., Nechaev,D. and Rost,B. (2021) Embeddings from protein language models predict conservation and variant effects. *Hum. Genet.*, https://doi.org/10.21203/rs.3.rs-584804/v1.
72. Hermans,A., Beyer,L. and Leibe,B. (2017) In defense of the triplet loss for person re-identification. arXiv doi: https://doi.org/10.48550/arXiv.1703.07737, 21 November 2017, preprint: not peer reviewed.
73. Kingma,D.P. and Ba,J. (2015) Adam: a method for stochastic optimization. arXiv doi: https://doi.org/10.48550/arXiv.1412.6980, 30 January 2017, preprint: not peer reviewed.
74. Finn,R.D., Clements,J. and Eddy,S.R. (2011) HMMER web server: interactive sequence similarity searching. *Nucleic Acids Res.*, **39**, W29–W37.
75. Pedregosa,F., Varoquaux,G., Gramfort,A., Michel,V., Thirion,B., Grisel,O., Blondel,M., Prettenhofer,P., Weiss,R. and Dubourg,V. (2011) Scikit-learn: machine learning in python. *J. Mach. Learn. Res.*, **12**, 2825–2830.
76. Sillitoe,I., Cuff,A.L., Dessailly,B.H., Dawson,N.L., Furnham,N., Lee,D., Lees,J.G., Lewis,T.E., Studer,R.A., Rentzsch,R. *et al.* (2013) New functional families (FunFams) in CATH to improve the mapping of conserved functional sites to 3D structures. *Nucleic Acids Res.*, **41**, D490–D498.
77. Peng,L., Oganesyan,V., Damschroder,M.M., Wu,H. and Dall'Acqua,W.F. (2011) Structural and functional characterization of an agonistic anti-human epha2 monoclonal antibody. *J. Mol. Biol.*, **413**, 390–405.
78. Himanen,J.P., Goldgur,Y., Miao,H., Myshkin,E., Guo,H., Buck,M., Nguyen,M., Rajashankar,K.R., Wang,B. and Nikolov,D.B. (2009) Ligand recognition by A-class eph receptors: crystal structures of the epha2 ligand-binding domain and the epha2/ephrin-A1 complex. *EMBO Rep.*, **10**, 722–728.
79. Webb,E.C. (1992) *Enzyme Nomenclature 1992. Recommendations of the Nomenclature committee of the International Union of Biochemistry and Molecular Biology*. 1992 edn. Academic Press, New York.
80. Sillitoe,I., Dawson,N., Lewis,T.E., Das,S., Lees,J.G., Ashford,P., Tolulope,A., Scholes,H.M., Senatorov,I., Bujan,A. *et al.* (2019) CATH: expanding the horizons of structure-based functional annotations for genome sequences. *Nucleic Acids Res.*, **47**, D280–D284.
81. Jensen,L.J., Gupta,R., Blom,N., Devos,D., Tamames,J., Kesmir,C., Nielsen,H., Staerfeldt,H.H., Rapacki,K., Workman,C. *et al.* (2002) Prediction of human protein function from post-translational modifications and localization features. *J. Mol. Biol.*, **319**, 1257–1265.
82. Nair,R. and Rost,B. (2005) Mimicking cellular sorting improves prediction of subcellular localization. *J. Mol. Biol.*, **348**, 85–100.
83. Kernytsky,A. and Rost,B. (2009) Using genetic algorithms to select most predictive protein features. *Proteins*, **75**, 75–88.
84. Dessailly,B.H., Nair,R., Jaroszewski,L., Fajardo,J.E., Kouranov,A., Lee,D., Fiser,A., Godzik,A., Rost,B. and Orengo,C. (2009) PSI-2: structural genomics to cover protein domain family space. *Structure*, **17**, 869–881.
85. Dallago,C., Schuetze,K., Heinzinger,M., Olenyi,T., Littmann,M., Lu,A.X., Yang,K.K., Min,S., Yoon,S., Morton,J.T. *et al.* (2021) Learned embeddings from deep learning to visualize and predict protein sets. *Curr. Protoc.*, **1**, e113.
86. Van der Maaten,L. and Hinton,G. (2008) Visualizing data using t-SNE. *J. Mach. Learn. Res.*, **9**, 2579–2605.