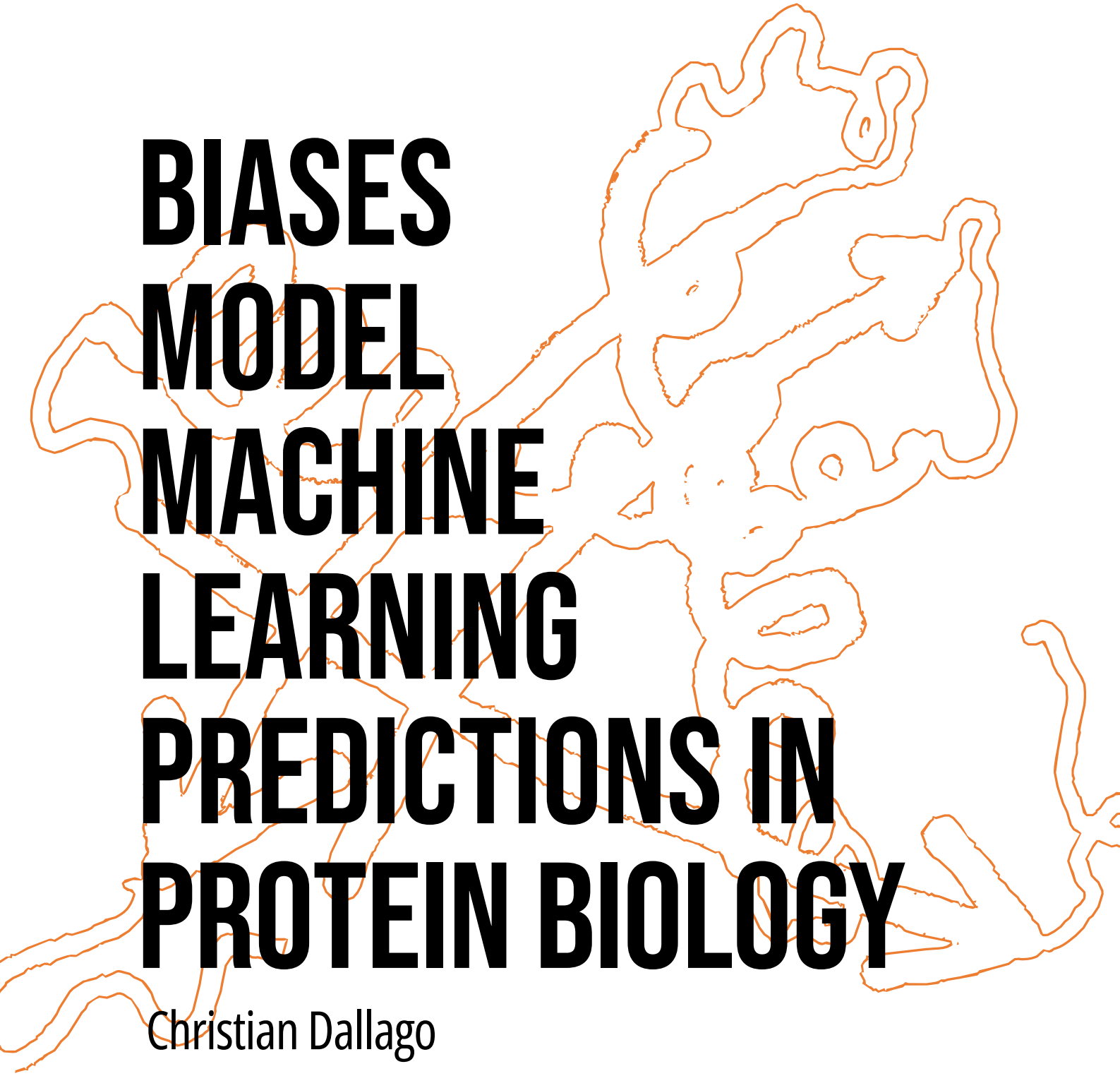


BIASES MODEL MACHINE LEARNING PREDICTIONS IN PROTEIN BIOLOGY

Christian Dallago

Technische Universität München
TUM School of Computation, Information and Technology

Biases Model Machine Learning Predictions in Protein Biology

Christian Dallago

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

Prüfende der Dissertation:

1. Prof. Dr. Burkhard Rost
2. Prof. Dr. Mathias Wilhelm
3. Prof. Anne-Florence Bitbol, Ph.D.

Die Dissertation wurde am 29.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 21.11.2023 angenommen.

ACKNOWLEDGEMENTS

Giving credit to the countless people that enabled me in producing the following pages is the entire reason for this manuscript to see the light of day, by and large for me to dwell in these endeavors, and probably amongst the hardest things *to get right* (rewrite n>100). Beyond the hardships, throughout which I could always count on helping hands, I recognize once again to have had the incredible privilege of being allowed to study what I wanted, where I wanted, in the time I needed; to have been allowed to fail, in science and life, sometimes quite spectacularly, without leaving indelible marks; to have been allowed to claim successes supported largely by others. As such, the work presented in the following is beyond a summary of years invested in scientific and personal trial and error. It serves as relatively miniscule token of appreciation compared to the immense patience, understanding and help that family, friends, and colleagues I relied on over the past *too-afraid-to-count-them* years have so kindly gifted me.

To Christoph Wüstner, Markus Schmidt, Elisabeth Neumann, Jan Kukačka, Christopher Göth, Alex Sobieska, Hannes Stärk, Daniel Graziotin, Céline Marquet, Michael Heinzinger, Roc Reguant, Stefano Trebeschi, Tim Karl, Maria Littmann, Yichun Lin, Maximilian Miller, Juan Miguel Cejuela, Guy Yachdav, Marcel Rodgers, Silvia Coran, Victorie Monnier, Deimos Larcher, Wessel Van Haarlem, Laura Venturato, Anna G. Green, John Ingraham, Jody Mou, Kadina E. Johnston, Andrea Schafferhans, Bruce Wittmann, Gary Bader, Luisa F. Jiménez-Soto, Tobias Piffrader, Nick Bhattacharya, Kevin K. Yang, Rostislav Nedelchev, Marco Munarini, Luca Rosalia, Verena Burger, Klaus Niedermair, Lucas Thomée, Augustin Luna, Hesam Rabeti, Timothy Lennon, Robert Feeney, Alexandra Franz, Moritz Moroder, Inga Weise, Konstantin Schütze, Sebastian Persson, Antonio Luca Alfeo, Mathias Wilhelm, Raian Eduardo Contreras, Marco Lanca Fischnaller, Patroklos Samaras, Jonas Reeb, Yassine El Alami, Jeffrey Wong, Chris Soon Heng Tan, Sami Afifi, Paolo Di Stefano, Tim Schlachta, Caner Hazirbas, Matthias Schmidt, Stefano Vaona, Daniele Parisi, Kordian Bruck, Nicolai K.H. Barth, Maximilian Denninger, Malte Butsch, Nima Dehmamy, Bernd Kohler, Vera Von Leon, Dinko Milaković, Yannick Mahlich, Natalia Quinones Olvera, Nicolas Fafchamps, Max Franz, Adam J. Riesselman, Nicolai Tornow, Peter Koo, Ananthan Nambiar, Ali Madani; to science dad Burkhard Rost, grandad Chris Sander, big sis' Yana Bromberg and Tatyana Goldberg; to the friends in Italy, Spain, the U.S., Singapore, France, Germany, Sweden, and wherever else life has brought you; to my former and current colleagues at Rostlab, Sanderlab and Markslab; to the collaborators; to my students; especially to my mom Patrizia, siblings Sarah & Alex, and grandparents Laura & Romano; to all who have contributed to my growth as a scientist and person, during and leading up to the long walk that finds a fork here:

thank you!

PREAMBLE

Probably one of the most common observations I read and wrote over the years of my doctoral studies was that "*protein structure/function are experimentally known for much fewer proteins for which sequence is known*". Before dwelling into what protein structure, function, or sequence are in the following chapters, I want to highlight here that this observation encodes one more crucial piece of information than may immediately be apparent. While we do know some experimentally validated aspects of proteins (structure, function and even sequence through mass spectrometry), and we can putatively produce more data for the protein language (sequences) based on *translation* rules (from DNA to protein sequences), what we observe overall in protein sequence databases, whether annotated or putative, is but a fraction of proteins that constitute the diversity of life. Here the potential diversity of protein sequences is to be seen not only as the biodiversity present at the current evolutionary timestamp, but throughout evolution, from millions of years ago to today. For instance, we don't have nearly as much sequencing data on viruses (making up 2% of protein sequence databases) as we do on bacteria (68%), and conversely, we often collect *representative* sequences (of a spices), rather than individual ones (from individuals). On top, much of what existed in the protein space will be lost in history, as time run its course and sequencing is a modern invention. Yet, *historic data* is desirable to build an understanding of how we came to be, or in biology terms, of evolutionary turns and twists, which are often needles (weak signals) in the most proverbial of haystacks (big data). Nevertheless, as a practical result, we are subject to biased data in biology, for instance by what protein sequences we store in databases, necessarily constrained by when they appear in evolution (mostly now, except some ancestral sequences found in fossils, permafrost, and soil). This becomes more tangible as we get into the *annotated* protein space, a tiny fraction of the recent snapshot of proteins recorded, where biases accumulate due to a mix of experimental limitations and feedback loops, ultimately imposing limits on our ability to experimentally model what proteins *do* (their function) and how they look in three dimensions (their structure).

In all of this, researchers operating at the interface of computers, statistics and biological data are tasked with producing first fundamental and then better ways of analyzing and interpreting available biological data to fill the knowledge gaps and smoothen selection biases from experimental approaches, somewhat like inventing and repairing a compass while navigating on the surface of the ocean with the goal of mapping the marine life below. Machine learning and software come in handy, providing a vast array of techniques to learn from available data to infer general patterns that could be leveraged minimally for coarse analyses, and often for predictive purposes. As such, *biases* do *model machine learning predictions in protein biology*, as we build tools on assumptions inferred by data available today, producing *computational representations* (e.g., the weights of machine learning models) of biology we can use to infer future aspects of proteins.

As towards the end of my doctoral studies I found the idea of biases in protein bioinformatics to be exciting, the following discourse focused on contextualizing some of my scientific contributions around how biases play a role in machine learning proteins, especially sequences, and how to push the boundaries through the curation of new datasets, new machine learning tools and software. A secondary personal goal was to perform a reflection on where we stand and what we must look out for to create fair and valid machine learning tools in protein biology, especially in the context of my own contributions. As a result, I found that contributions on representation learning for protein sequence offer, amongst other things, an attempt to smoothen selection biases of sequence databases by learning *general sequence representations* from large protein databases. Unsupervised approaches for protein function predictions, particularly the instances where function is not categorical (e.g., via ontologies) but rather left on a continuous, could be viewed as smoothening annotation biases by focusing on similar protein pockets (clusters) in a high dimensional space. Software solutions for the visualization, annotation and prediction of protein attributes can be viewed as attempts to push scientific explorers out of uninformative feedback loops, as well as to make science more accessible to all. Ultimately, while biases play a role in machine learning biology and may sometimes sidetrack our understanding of the biological world, the models of biology we can produce have many advantages. For instance, we can use machine learning in biology to drive development of cancer therapeutics, but we should remain vigilant that most of our sequencing data, including from cancer patients, comes from specific pockets of the global population, which could limit the applicability of our discoveries to certain individuals. However, it's worth considering that in the absence of *any* solution, especially when dealing with disease, *every* solution is a step forward.

In closing, let me highlight that in writing this dissertation my primary goal was to make it engaging (*fun* is not quite scientific), based on science dad's (see Acknowledgments) "*science is communication*" mantra. (Self-perceived effective) communication in this instance proved a monumental task requiring a year of nurturing and two weeks of labor to bundle the essence of about 20 manuscripts (on sometimes entirely different topics) into a single, cohesive body of text. Little did it help that this piece needed to fulfill all the (often quite tedious) constraints imposed by *regulatory bodies*. As a result, unsurprisingly to those who know me from the trenches of early drafts, I chose to a) *politely* rebel against the system, and b) fall short on nitty-gritty insider details to give way to big picture ideas, which should hopefully facilitate even the least seasoned bioinformatics research reader in engaging with the topics discussed, while also hopefully not drawing on the ire of *regulators* to fault the attempt. If you come across terminology you don't know, my suggestion is to just move on – it's not about the detail! While for those that fancy numbers, graphs, and hardcore terminology, I highly suggest you skim over the manuscripts mentioned in the "*Scientific contributions*", all of which are open access and linked.

1 Summary

Annotations of protein function and structure are available for far fewer protein sequences than those reported in protein sequence databases, which in turn are far fewer than all proteins that have existed in nature thought history. Bioinformatics is tasked with leveraging the limited information encoded in protein function, structure, and sequence annotations to find and extract general knowledge about biology which practitioners can leverage to develop solutions improving the state of living things. Evidently, using little data to extrapolate blanket interpretations of biology is a delicate practice that could be tipped towards unfavorable outcomes by selecting information, either consciously or unconsciously. This thesis is an attempt to review the state and origin of biases in protein bioinformatics, as well as a perspective on how applications of machine learning and software could be used to address them. In practice, I will frame several efforts I contributed to in- and around machine learning to convey computational meaning to human collected protein data, and to translate machine predictions to meaningful human features through software. The outcome of this deep dive will highlight that representation learning approaches on large protein sequence sets could smoothen biases induced by curated sequence datasets and limited supervised function/structure sets. Furthermore, the continuous and space-sharing nature of these representations allows them to be correlated to protein function on a continuum, potentially overcoming limitation of sharp protein function categorizations. Finally, software democratizing protein predictions, for instance through visualizations, provides means to dissect machine models of biology potentially enabling new discoveries.

2 Introduction

In the following sections, I will introduce some aspects of protein bioinformatics and how biases may determine experimental annotations. These are fundamental concepts such as what proteins are, how they function, how machine learning comes into play to learn their properties, and how humans and machines exchange models of biology.

2.1 Proteins: function, structure, and sequence

Proteins are the building blocks of life. They carry out functions within and outside cells, tissues, organs, organisms, and spices, as messengers and messages (Ramilowski *et al.*, 2015). The common process of protein generation is started when genetic information encoded in genes (DNA) of an organism's cells get transcribed into messenger RNA (mRNA), which is then translated by cellular machinery into proteins. The three macromolecules involved in this process (DNA, RNA & proteins) all deserve credit for the diversity and complexity of life, they each occupy physical space in unique three-dimensional shapes, are composed by smaller parts (from bases or amino acids down to atoms) and carry out a multitude of functions sensitive to their operating contexts. Building a discourse around these biological entities requires picking a point of view and accepting some assumptions, ultimately conveying a mental model to you (the reader), which conversely will be the model used to interpret biology in most of the applications discussed in the following pages.

One possibility to model biomolecules is to look at them through high-resolution microscopy pictures of cells in tissues (Schermelleh *et al.*, 2019), which in the ideal case can be used to frame DNA, RNA, and proteins as they appear in cells. The prerequisite here would be that we can get high-resolution pictures of these macromolecules from tissues, and the following assumption is that these pictures suffice for all kinds of predictions we are interested in. However, we unfortunately are still away from pictures at the required resolution to operate on biomolecules purely through imaging (Schermelleh *et al.*, 2019). Thus, the preferred representation in bioinformatics, and relevant for the discourse here, are proteins (and RNA, and DNA) as ordered strings (like written sentences) of amino acids. Here the prerequisite is that each of these biomolecules can be further divided into sub-parts (bases and residues), a condition we are able to satisfy, and the assumption is that these strings, in a particular order, represent the entirety of the *meaning* of that macromolecule. This assumption is also largely satisfied, although the reality of biology is complex, and rarely is it the case, as we will see shortly, that a protein functions without acting on some other entity. As such, while sequence does carry much of a protein's meaning, and as we'll see, we can reconstruct a lot of information from it, quite some meaning is

also dependent on contextual factors, such as by the cellular environment (in which tissue and at what time of the cellular cycle is the protein expressed) (Dobson, 2003).

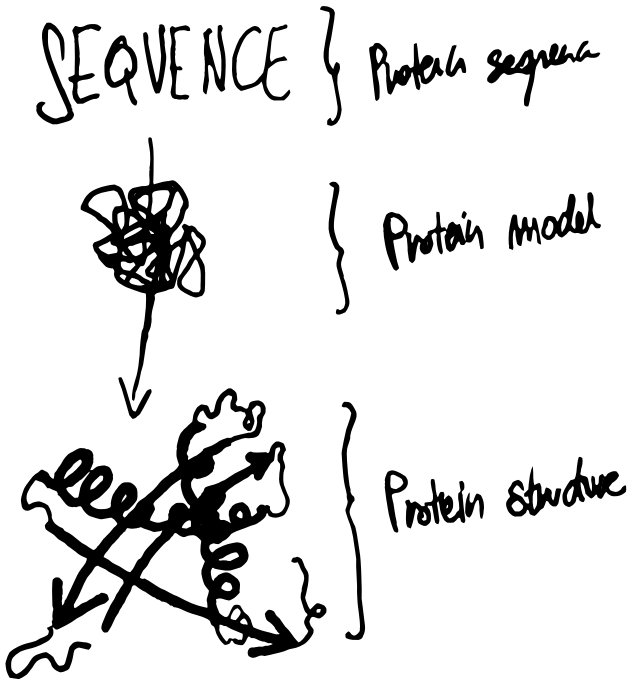


Fig. 1 – Models of protein sequences. *Protein sequences contain multidimensional information that can be encoded in an abstract form (e.g., computational protein models), which can be leveraged to reconstruct protein properties (e.g., structure). Effectively, these protein representations are modelled by protein data available to machine learning methods.*

The three fundamental biological macromolecules DNA, RNA and proteins interpreted as text could be put in an analogy to Old English, English, and Mandarin, respectively. DNA and RNA are to some extent similar, both can be represented in written form, and each sentence or document written in the DNA/RNA language carries some meaning. Proteins, on the other hand, while still forming sentences and documents with structure and meaning, differ in some fundamental respects from DNA/RNA, for instance by encoding more compact representations of meaning (in the analogy, what ideograms are to Arabic characters). While the analogy to variations of English and Mandarin is an oversimplification (for instance some of the DNA/RNA language can simply not be translated to proteins), fascinatingly, all these biological languages share a communality to natural language in that they can *represent* physical objects, like a chair, simply through the chaining of characters in an orderly fashion. In fact, natural language and evolution gives you the ability to picture a chair just from reading five tokens on some digital or physical medium (provided you know the language and you can read, neither of which are trivial prerequisites). Thus, from an information perspective, what we will operate on is the language of biomolecules with the intent of constructing meaning from it, the same way we construct meaning from natural language. From a practical perspective, this dissertation will focus on proteins, rather than DNA or RNA. The

following subchapter will expand on complex attributes of proteins (*function*), moving to more tangible ones (*structure*), to finally end up with the most basic protein property, *protein sequence*.

2.1.1 PROTEIN FUNCTION

Protein function is probably the most elusive protein property to define. A protein can perform function(s) self-sufficiently (see *catalytic enzymes*), or in combination with other molecules (see *protein complexes*) (Ashburner *et al.*, 2000). Proteins may perform different functions at different timepoints of their existence in the cell (Henzler-Wildman and Kern, 2007), they may *lose* or *gain* functions through evolution as could be seen today by sequence/structure similar proteins in different organisms doing different things (Barua *et al.*, 2021). Functions may be trivially measurable (protein X functions in *cytoplasm*), they may be relative to a protein's role in a biological pathway (protein X activates transcription of Y) or be characterized through far-reaching complex phenotypes like disease (mutations of protein X causes disease Z). On top, many aspects of protein function are most likely on a continuum, rather than ON/OFF toggles, meaning that their activity can be modulated to higher or lower output. This could be achieved through contextual means, for instance, by having *more* or *less* of a protein in a cell (thus regulating protein expression) (Woods and Vousden, 2001), or by intrinsic properties of proteins, such as changes in sequence (Fowler and Fields, 2014; Yang *et al.*, 2021).

To standardize discourse around what proteins do / how they do it, researchers introduced categorization schemes and ontologies. At the fundamental biological level, i.e. describing the protein rather than its role in complex phenotypes (disease) or high order interactions (pathways), one of the better known and maintained bio ontologies is the Gene Ontology (GO) (The Gene Ontology Consortium, 2019). GO comes with an associated database that maps annotations to proteins in the protein sequence database UniProt (The UniProt Consortium, 2021), namely the Genene Ontology Annotation (GOA) database. Through GO, proteins can be categorized on three major axes: the molecular function (MFO) a protein is involved in (e.g., *signaling*), the bigger biological processes (BPO) it is involved in (e.g., *DNA repair*), and the cellular compartments (CCO) it functions in (e.g., *nucleus*). Next to GOA stand resources that look at one or the other aspect of function in greater detail, or under a different light. Case and point: Enzyme Commission (EC) numbers and the ENZYME database (Bairoch, 2000). Enzymes are proteins involved in catalyzing chemical reactions, which in turn regulate processes in- and outside cells. Enzymes also play a fundamental role in signaling (Mildvan, 1997). While potentially serving similar purposes (considering the MFO and BPO ontologies in GO), GOA (The Gene Ontology Consortium, 2019) and ENZYME (Bairoch, 2000) differ widely in the numbers of annotations, from the 944'228'169 reported by the former, to the 6'553 reported by the latter as of February 2021. This disparity can be attributed to several reasons, the main one probably being that the GOA number reported accounts for "annotations via sequence similarity" and even computational predictions,

implying that instead of having some physical experiment in a lab on dishes or animals resulting in an annotation with a degree of certainty and "reality", we use expertise and machines to infer annotations. This may be alright if the prediction error is within the experimental error, unfortunately this is rarely ever the case in biology, especially when operating on coarse annotations such as protein function, and the experimental error may not be fully captured in the models machines compute on.

A further example of function annotated by many is protein subcellular location (i.e., where a protein locates in the cell). Other than GOA's CCO annotations (The Gene Ontology Consortium, 2019) there are SwissProt (The UniProt Consortium, 2021), the Human Cell Atlas (HCA) (Thul *et al.*, 2017), and PROLOCATE (Jadot *et al.*, 2017), just to name a few. These resources, although conceptually recording the same data (protein subcellular location), differ by many aspects. First, they may use different ontologies if any ontology at all (case: PROLOCATE). This, in turn, can have an impact on what they record and what they decide not to record, e.g., in PROLOCATE, only eight subcellular localizations are considered, while we know many more exist in nature. Compare this to the 2'702'774 possible GO terms in the CCO ontology (fair: some may be related as GO has a "tree" structure), for which there exist 222'477 entries in GOA having both CCO annotations and being at least experimentally validated. Secondly, different datasets may be selection biased by the experimental assays used (case: immunofluorescence in HCA vs. isobaric labeling in PROLOCATE), which come with tradeoffs and favor some conditions/cells/locations over others. Third, these datasets may be further selection biased by recording "*interesting*" organisms (case: human in HCA and mouse in PROLOCATE) limiting the space of "annotated" to only a few, potentially similar organisms. On top, annotations in these sets may contradict each other (Marot-Lassauzaie et al., 2019). Ultimately, all this *uncertainty* begs the question: if a researcher in bioinformatics is interested in predicting subcellular location for proteins from sequence, what will, at this point, the best dataset be? What will the true label be? Or, in fact: the biologically correct label (if in set A protein X is in cytoplasm and in set B protein X is in nucleus: which one is it?) My conclusion: it depends. If the researcher is interested in predicting where human proteins locate, then HCA's dataset may be the best for validation and testing. Yet, given scarcity of data, PROLOCATE data may be used for training, as many proteins in human and mouse share a certain level of sequence similarity (implying they will most likely do the same thing, more later). If the researcher's goal is absolute generalization, sequence similarity and "in-distribution" predictions are in fact hurting, so the best way to go about this is to cluster protein sequences and train on a certain cut out of the sequence space, and validate and test against other, non-overlapping cuts of the sequence space. For this particular use case (but so many others) the complexity that arises is what to do with sequences falling in a cluster having different, maybe even contradicting annotations.

Deep mutational scanning (DMS) assays (Fowler and Fields, 2014) are an attempt to describe function on a continuum. In these experiments, a particular function (e.g., binding to another molecule) is quantitatively measured under sequence changes (most often by substituting

iteratively every residue in a known protein sequence to every other possible alternative). The usual DMS set will minimally contain 19*sequence length samples each one Manhattan distance away from the starting sequence. While giving a detailed picture of the functional impact for the immediate sequence neighborhood to some reference protein sequence, these sets are still very few and noisy (Wittmann *et al.*, 2021; Reeb *et al.*, 2020). For instance, they are biased to the activity/function measured, and may thus not fully capture the protein's ability to perform some different activity it is involved in during its lifetime.

Fig. 2 – Proteins can have multiple functions. *If functions are on a continuum, then wildtype (as found in nature) proteins may be intersections of functional states at an optimum (e.g., f1 is binding to protein X, f2 is thermal stability and f3 is ability to migrate to the nucleus). Tweaking the protein sequence could enable moving along one function or multiple functional hyperplanes.*

Another contributing factor to ambiguity, complexity and error in protein function annotation is driven by cumbersome processes to move high-resolution/high-accuracy annotations between humans and machines. On the one hand, from humans to machines, much of the data collected by scientists through experiments is summarized in text in scientific manuscripts that are difficult to use for computations. For example, in the case of pathway data, which represents how complex biological processes happen, human curators extrapolate data from manuscripts and translate them to computable artifacts (e.g., the BioPAX (Demir *et al.*, 2010) format) in efforts like Reactome (Jassal *et al.*, 2020) and KEGG (Kanehisa *et al.*, 2021). On the other hand, from machines to humans, predictions are seldom accompanied by useful software to navigate machine learned models. This becomes especially challenging when dealing with multimodal annotations of protein functions, which may be better contextualized by complex visualizations layering different biological dimensions.

Computational tools offer, on the one side, solutions to model protein function (chiefly, here, machine learning) by smoothing potential biases and contradictions from labeled sets, and on the other side, software to communicate findings from and to machines (Littmann, Heinzinger, Dallago, Olenyi, *et al.*, 2021). Trivially, a biologist could predict subcellular localization given a machine learning device with weights adjusted on a particular dataset using a particular machine learning architecture. More fundamentally however, that same machine learning device with those weights is a model of what subcellular localization is, it captures a *meaning*, conditioned by several assumptions, and may account and adjust for inconsistencies and biases in annotations (be that: sources of annotation, labels used, or experimental error). Software tools to explore predictions beyond outputting text, but by providing engaging displays of information to users may facilitate scientific discoveries and break feedback loops of predictions feeding experimental annotations, in turn generating more diverse datasets for new predictive applications.

2.1.2 PROTEIN STRUCTURE

As opposed to function, protein structure can be derived by first principles, and thus describing what structure is (at different levels) is simpler. However, also for protein structure, complexity of measurements and limitations of experimental approaches curb the space of possibilities, resulting in some degrees of experimentally induced selection biases.

The highest-fidelity representation of protein structure is through 3D coordinates of atoms making up said protein. A relatively big collection of these is available at the Protein Data Bank (PDB) (Berman *et al.*, 2000). 3D protein structures get recorded mainly through one of three types of experiments, whose setups determine what kind of proteins can be measured and at what resolution. These are important factors: high-resolution structures are necessary to understand dynamics of how proteins interact with other molecules (e.g., other proteins, DNA, RNA, or *small molecules*), but so is being able to measure proteins in their native environment (e.g., the protein's "fold" in the cell membrane). However, as is unfortunately often the case in biology, "needs" often clash with reality of our tools and techniques. X-ray crystallography offers high-resolution protein structures (making up ~88% of PDB as of March 2021), but experimental preparation disrupts the native environment of proteins, and per effect, may introduce unwanted artifacts and, even worse, limit the types of proteins that can be measured. For instance, in this modality, membrane proteins, which play fundamental roles in disease, cannot be characterized in their native environment. A lower resolution technique, NMR (~8% of PDB), offers instead to capture proteins closer to their natural context, at the cost of often prohibitively low resolution for most applications (e.g., to study protein-small molecule interactions). Hybrid approaches merging these techniques may lead to the best of both worlds for some proteins (Ottmann *et al.*, 2007), but are not widely adopted. Similarily, a promising in-between (high-resolution and native context) technique exists (cryoEM), yet due to its relative novelty, and only recent limited breakthrough in

resolution (Yip *et al.*, 2020), not many proteins have been characterized using cryoEM (~4% of PDB), let alone: at high enough resolution.

To add insult to injury, while it may be tempting to believe that although there are experimental limitations, we may *just* measure any protein that we are interested in, most experimental techniques rely on template structures to seed reconstruction of measurements, i.e.: one can measure things that bare some similarity to known structures or on structures we can predict. This may constitute at best another selection bias, and at worse a confirmation bias. Furthermore, all traditional, non-predictive experimental techniques are resource intensive (in equipment and time), and their cost increases with higher starting uncertainty (and repeat measurements). Naïvely picking a protein of interest and starting a campaign to determine its structure from scratch is thus doomed to make life much harder, a reality that many PhD students and doctoral advisors avoid for a variety of reasons, many times rightfully beyond the boundaries of scientific interest or relevance. However, these limitations translate to sets containing many redundant structures, often from similar protein families/folds. Going back to why, thus, the PDB is *relatively* big: while many structure are deposited, usable are those with high resolution (or minimally $\leq 3\text{Å}$) which as of October 2020 are 225'161; however, many of these are redundant (meaning: similar structures / similar proteins), and if we were to reduce to proteins with sequence overlap below 20 percent sequence identity (PIDE), then we end up with 8'988 structures. While this is a good number, an even better Å cutoff, especially when looking at contacts, is 1.2Å, which will result in a usable set of 678 structures – a very low number in deep learning terms.

Depending on the goal of the analysis redundancy is not always unwanted. If the redundancy is at the level of protein sequence (several structures exist for similar protein sequences in different contexts), and the goal is to model structure dependent on e.g., binding to other molecules, then two folds (structures) for the same protein sequence may come in handy. For instance, the now infamous spike glycoprotein (S) of SARS-CoV-2 has two main conformations: one when resting on the surface of the virus, and one when the virus is binding through the Spike protein to the ACE2 receptor on the surface of, amongst others, human cells (O'Donoghue *et al.*, 2021; Song *et al.*, 2018; McCallum *et al.*, 2020; Henderson *et al.*, 2020). Having various models of proteins sharing sequence similarity to the Spike protein will help model the different conformations it may assume. Other than "rigid" conformational changes (analogy: an open or closed door), there are conformational changes of energy unstable parts of the protein (also, *disordered regions*) that often have a functional meaning (analogy: shoelaces that when tightened and knotted hold the shoe in place, and when left to their own devices just wobble around seemingly without purpose). For disordered regions in proteins, as in the previous case, having redundant structures might give a hint as to how those regions behave under different conditions (e.g., in a complex), and more challenging attributes, such as how the protein functions. Overall, conformational changes suggest a link between the mechanics of structure and the purposes of proteins encoded by their function, and thus, at least to some extent, structure has an influence on (and a predictive ability for) protein function.

A lower resolution representation of structure is via residue-level annotations of *secondary structure*. Best known through the Description of Secondary Structure of Proteins (DSSP) method *(Kabsch and Sander, 1983)*, protein sequences can be annotated at the residue level by observing in what conformation each residue participates in the folded 3D structure of the protein. The three most prominent secondary structure classes are alpha-helices, beta-bridges (that can form sheets and strands), and whatever is neither alpha nor beta (commonly *other*, not to be confused with *irregular*, which more closely resembles *disorder*). This different protein structure representation, and its ability to more closely couple with protein sequence, provided a useful proxy in lowering the complexity of structure reconstruction for predictive purposes, although lowering resolution tremendously.

Machine learning approaches on protein structures have a lot of potential. Differently than protein function predictions, the protein structure task allows to define some objectives with undeniably clear solutions, based on geometry and physics. For instance, one could consider only the "native" state of proteins (e.g., after assembly, not involved in any activity), and ask the question: what is the secondary structure of native proteins? Complexity can be dialed up, and one could attempt to predict the whole 3D protein structure. In fact, a solution touted as *the solution* to this task exists, on paper since December 2020, and available to all since early summer 2021. AlphaFold2 (Jumper *et al.*, 2021) made headlines by offering to predict protein 3D structure at astonishing accuracy (sometimes better than experimental assays), validated by a competition that was started to prepare test sets with a degree of "uniqueness" of the structures, to promote generalization and tackle bias. The full range of tasks that can benefit from *AlphaFold2*'s predictions remain to be fully uncovered, but recent preprints suggest beneficial use for disorder prediction (Jumper *et al.*, 2021), mutation effect prediction (Pak *et al.*, 2021), and more (Akdel *et al.*, 2021; Modi and Dunbrack, Roland L, 2021).

2.1.3 PROTEIN SEQUENCE

The most straightforward representation, and conversely property, of a protein is its sequence. Proteins can be broken down to ordered chains of residues, each representing one of 20 amino acids. This ordered sequence of residues can be written in text form, and is the starting point of most bioinformatics approaches, especially those trying to "*de novo*" predict aspects of proteins.

A protein sequence uniquely identifies a protein, and the same protein sequence will uniquely *fold* into the same native 3D shape, i.e. protein structure. As protein sequence is thus uniquely linked to protein structure, and protein structure is linked to protein function, there is a link between protein sequence and function (Rost, 1999; Nair and Rost, 2002). Cutting a few corners, we may assume that protein sequence encodes all the information necessary to characterize a protein (structure and/or function) up to contextual factors (such as: which cell does the protein express in? = different subcellular locations? Is it secreted? = subject to varying ph levels? Is it

binding? = conformational changes?). In fact: from a physical perspective, considering all the forces atoms forming a protein/its residues are subject to, it is possible to accurately reconstruct protein 3D structure through simulation. However, while this is true in theory, in practice the complexity of considering all the physical constraints of protein sequences growing in length quickly outpaces the amount of compute available, rendering exact simulations of proteins infeasible (similarly as our inability to precisely model the weather), even in the polypeptide (up to 30 residues) space. Nevertheless, we can try to approximate/predict structure/function by using probabilistic (Ingraham *et al.*, 2019) or frequentist (Hecht *et al.*, 2015) approaches, and even simple pattern matching (Lange *et al.*, 2007) just from an input sequence enhanced by evolutionary information leveraged through clever lookups (how did the protein evolve in time/throughout species?) or machine learned sequence representations.

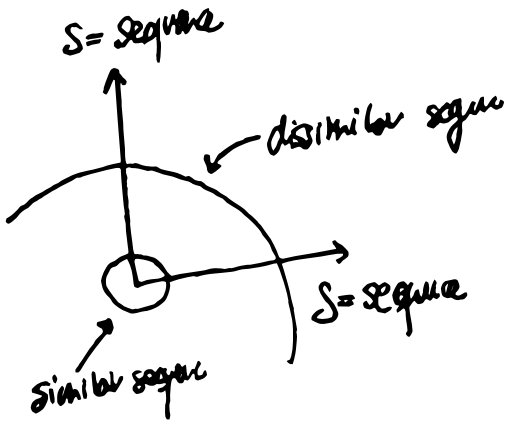


Fig. 3 – Sequence neighborhoods contain similar functioning sequences. *In a sequence-sequence plane, those sequences within a circle of small radius share sequence identity and are thus more likely to fall into similar 3D shapes and perform similar functions, while those in larger circles are likelier to be more dissimilar in structure and function.*

Starting point for these approaches are thus protein sequences. Quantitatively, proteins can be assayed experimentally via proteomics (Samaras *et al.*, 2019), answering the question "How much of a *known* protein sequence is present in a sample?". But for the question of "what's the sequence of a/any protein in a sample?", limited options are available (Howorka and Siwy, 2020). One assay recently showing promising results for the protein space is nanopore protein sequencing (Howorka and Siwy, 2020), but as of today: it is limited to small exploratory studies. In the absence of high-throughput solutions operating directly on proteins, the accepted solution is to exploit the central dogma of molecular biology (Cobb, 2017): DNA (a blueprint of the machinery) transcribes RNA (an interpreter) which translates proteins (the machinery). This dogma can be leveraged by sequencing DNA/genes to translate them into protein sequences. As mentioned in an earlier section: from an information content / text perspective, DNA and RNA are interchangeable, both composed of a vocabulary of four characters (n_{DNA} = {C, G, A, T}; n_{RNA} = {C, G, A, U}; n meaning "nucleotide"), of which three are identical and overlapping (C, G, A) and the fourth can be directly

mapped between representations (T = U). Protein sequences have a richer vocabulary ($res_{protein}$ = {A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V}; *res* meaning residue). For many triplets of DNA, which encode RNA nucleotides there are translation to protein representations (e.g., DNA({UGU, UGC}) = protein('C')), as well as nucleotide triplets that signal start (DNA(AUG)) and end (DNA({UAA, UGA, UAG})) of a potential protein. At this point it's worth noting that, as much as these translation rules may be convenient, they introduce several levels of ambiguity (e.g., DNA(AUGUA) = protein?). Furthermore, biological mechanisms may determine only parts of a protein to code, or genomic segments to encode different proteins using the same genetic material (spicing). Truly here, the hacks that biology uses to decode genomic material into several variations of proteins appear to be endless (or, at least, not entirely characterized). For instance, viruses such as SARS-CoV-2 can leverage several mechanisms to code one or the other protein at a given point in time, as well as variations of proteins by mixing the same genetic material in different ways (O'Donoghue *et al.*, 2021). On top of this, while exact, expensive DNA sequencing techniques exist, the high throughput "next generation" (NGS) techniques that allowed to go from $100mio to sequence a human's genome in the early 2000's to merely $1k today come with some perplexity regarding the correct reconstruction of the genome (DNA) sequence, and, per effect, translation to protein sequences. TrEMBL is a part of the UniProt database (The UniProt Consortium, 2021) that contains *putative* sequences (as of April 2021: 214'406'399 sequences), obtained by applying the translation rules from nucleotides to proteins on *reference genomes* of organisms. "Reference genomes" themselves are single artificial genome assemblies of an organism obtained from processing and "averaging" individual genomes to minimize sequencing error and maximize reconstruction accuracy, artificially disregarding some level of natural variation found in life, more cleverly captured with pan-genomes (Tettelin *et al.*, 2005). While useful, reference genomes may introduce the same issue found when describing a group of people by their average height, i.e., there may exist no individual of average height. Conversely, by smoothing the perceived "measurement" error, we may instead smoothen natural variation.

Once experimental evidence for some aspect of a putative protein becomes available, e.g., by running a proteomics scan on a cell line for that organism, these no-longer entirely putative sequences become part of SwissProt (as of April 2021: 564'638 sequences), another section of UniProt (The UniProt Consortium, 2021). In proportion, for every protein sequence with some level of experimental annotation in SwissProt, there are 380 sequences for which we know only putatively (maybe!) that that sequence exists in that organism as deposited in TrEMBL, a striking 1:380 ratio, or 380 times sequences with unknown function, structure, and sequence for every one sequence with some level of either sequence, function, or structure certainty. These putative sequences highlight that beyond a hypothesis, the amount of "unknown" vastly overshadows the universe of "known". To complicate things, while apparently many sequences are deposited in UniProt (the primary protein sequence resource), whether they be putative or labelled through some experiment, they still come from a limited set of organisms for which we started to build reference genomes, few of which are completed. Most of these genomes are those of human-

relevant species, like human-comparable organisms (in terms of genome and proteins, e.g., mouse, pig, etc.), or of human-harming organisms, such as bacteria, viruses, or of organisms with potential industrial applications for human health, such as some plants. Efforts addressing the human-centric bias in sequencing exist, e.g., by metagenomic sampling (Steinegger and Söding, 2018), but without further labelling and interpretation of their outputs, these approaches will only scratching the surface, given the limit imposed by requiring reference genomes to reconstruct fragmented genomic reads, as well as by missing detail on the hacks used by underrepresented organisms (e.g., viruses) to code for proteins.

2.2 MACHINE LEARNING PROTEINS

Fundamentally, a desideratum in protein bioinformatics is to go from protein sequence (cheap to obtain and wildly available) to properties of proteins, such as atomic coordinates of the folded protein. This case may sound like a trivial problem to solve, considering that we know the physics behind amino acids and could model their interactions in space through simulations. However, this process is exponentially computationally expensive as proteins increase in sequence length and is not a viable option for the average protein of 350 residues (=amino acids). Frequentist based machine learning approaches, such as Deep Learning, offers an alternative to resource demanding simulations by approximating results to an optimum solution learned by observing known samples. Particularly interesting in this space are those methods that can model proteins from their sequence "*de-novo*", i.e., without any other knowledge but sequence. Predictions of protein structure have driven translational innovation in machine learning, software engineering, data science and biology for decades (Rost and Sander, 1992, 1993a, 1993b, 1994). Today, AlphaFold2 (Jumper *et al.*, 2021) promises to bridge the annotation gap from sequence to structure for a significant space of otherwise experimentally uncharacterized proteins. While tempting to believe that solutions like AlphaFold2 are an eureka moment, it's useful to consider that these tools were the culmination of decades of interdisciplinary research which drove four major advances:

1. Impressive engineering, from hardware (GPUs and networking) to software, connecting and speeding up every aspect of an end-to-end tool (so much so that it can be run from the web through cloud solutions (Milot Mirdita *et al.*, 2021))
2. Identification of structure-relevant input features, in particular "direct evolutionary information", first as background information (BLOSUM) (Henikoff and Henikoff, 1992), then through alignment of similar protein sequences in multiple sequence alignments (MSAs), to direct coupling analysis (DCA) (Morcos *et al.*, 2011), and finally to self-supervised learning through machine learning (Jumper *et al.*, 2021)
3. Advancements in machine learning, in particular convolutional networks and deep learning (AlQuraishi, 2021; Wu *et al.*, 2021; Ching *et al.*, 2018)

4. Growth of sequence and structure databases, enabling more depth (deeper alignments) and width (effect of sequence variation on structure) (The UniProt Consortium, 2021)

One cannot really disentangle the interleaving of machine learning research and biological discoveries which led to the incredible number of solutions available today for an array of applications. It's also apparent that whenever one field unlocks a new advance, the others have a unique chance to fill the void of opportunity. For instance, when DCA made its way in 2011 (Morcos *et al.*, 2011) revolutionizing the way we predicted protein structure from sequence (Hopf *et al.*, 2012), it took until 2017 for its application to be translated to mutation prediction (an aspect of protein function), as more complete mutational landscapes were needed (Hopf *et al.*, 2017) (a case of biology catching up with machine learning). It took some more years for machine learning, hardware and software design to catch up, in order to render this solution scalable (Frazer *et al.*, 2021) (an example of software engineering and hardware catching up with machine learning), and until curated datasets for assessment would become available (Dallago, Mou, *et al.*, 2021) (an example in dataset engineering catching up with biology and machine learning).

Machine learning protein structure can have an influence on predicting protein function as well, as may transpire from reading in-between the lines of previous paragraphs (Sander and Schneider, 1991; Nair and Rost, 2002; Rost, 2002). In fact, if we accept that proteins fold into definite 3D structures constrained by their sequence (and their cellular context), and if we accept that proteins share function when they fold in similar shapes, then we can assume that the structure of a protein largely influences its function. Predicting protein structure is thus a great proxy to some aspects of function as showcased by techniques that work for structure prediction (e.g., direct coupling analysis (Thomas A Hopf *et al.*, 2019) predicting effect of mutations on protein fitness (Hopf *et al.*, 2017)). Conversely, basing function prediction purely on approaches that work for protein structure may bias or constrain our approaches, and we could instead devise solutions to directly predict protein function from sequence using machine learning (Littmann, Bordin, *et al.*, 2021; Littmann, Heinzinger, Dallago, Olenyi, *et al.*, 2021), allowing these tools to highlight different realities than those biased by structure might (Meier *et al.*, 2021; Marquet *et al.*, 2021).

2.2.1 Generating computable representations of protein sequences

Machine learning protein sequences falls under the general category of "representation learning" (Bengio *et al.*, 2013), in particular answering the question "how to teach machines models of protein sequences", often irrespective of the *downstream* task these models might be used for (Bepler and Berger, 2019; Heinzinger *et al.*, 2019; Alley *et al.*, 2019; Meier *et al.*, 2021; Elnaggar *et al.*, 2021). Representation learned systematically (i.e., on all proteins, irrespective of organism or tissue) were initially motivated by the observation that successful machine learning tools in protein predictions where subject to two major limitations driven by the most import input feature

(evolutionary information). First, evolutionary information in the quality required by machine learning applications is unavailable for most proteins. For instance, MSAs are informative when they are deep (containing many sequences), as well as diverse (highlighting what shuffles of residues still render viable proteins within the overall context of a protein sequence) yet meeting both requirements is frequently not possible, especially for dark proteins, making up large chunks of the recorded sequence space (Perdigão *et al.*, 2015), let alone the not-recorded sequence space. Second, searching for evolutionary information explicitly (or, as I dubbed it earlier "direct evolutionary information") is increasingly computationally expensive, as the growth of sequence databases outpaces the increase of transistors in microchips (in other words: we can't rely on faster chips to search larger sequence databases). As such, models learning purely from sequence without relying on transformations on top (e.g., MSAs) are a desideratum since several years in bioinformatics. Thankfully, groundbreaking advancements in natural language processing (NLP) (Devlin *et al.*, 2019; Raffel *et al.*, 2020) set the stage, allowing machines to learn semantics of proteins just from observing the grammar encoded in protein sequences into protein language models (pLMs). Without going into the details of the models (which are explained in detail in the open access manuscript referenced in the "*Scientific contributions*" chapter), the common approaches used to learn meaning from sequence can roughly be partitioned into two sets:

1. Models that learn to predict the next residue in a protein sequence given the information from *previous* residues with some decay in length (i.e., the further in the past some residue, the less influential for the next prediction). These approaches relied on recurrent networks using Long-Short Term Memory (LSTM) modules (Heinzinger *et al.*, 2019; Alley *et al.*, 2019; Bepler and Berger, 2019). For instance, a model like this would learn what follows an ordered sequence of *tokens* such as "I have to run to catch the ____", by placing more attention (for the machine learning specialist: not in the machine learning "attention" sense) to the last word before the gap. It learns representations iteratively, meaning it first predicts the first word based on the fact that it's the first; then the second based on that the previous one was an "I"; then the third based on that the previous was "have" and some residual information about "I", and so on.

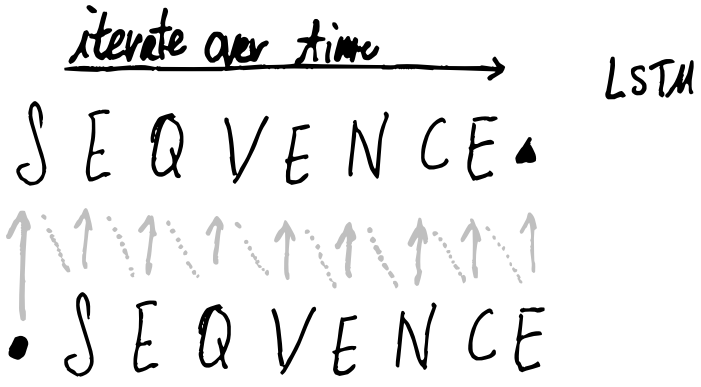


Fig. 4 – Recurrent Neural Network learning task. *In the bottom is the input sequence, on the top is the predicted sequence. A gray arrow upwards represents the prediction from the model based on the input below. ● represents the beginning of a sequence, while ▲ represents the end. At each step (left to right) the machine learning device is tasked with predicting the next character. Once a prediction is completed, the residual information from that step is carried over to the prediction of the next character (horizontal gray lines).*

2. Models that learn to reconstruct corrupted sequences based on the non-corrupted input (based on transformers (Devlin *et al.*, 2019; Raffel *et al.*, 2020)). For instance, these models would take a sentence and mask some words at random positions, such as "Today it's really _____ at the beach.". Naturally, you would prioritize a few words (sunny, windy, warm, etc.) over the many possible options in your head. The model is tasked with doing the same, in particular it is tasked with predicting the single correct word for the sentence from its uncorrupted form. In this case, there's no iterative learning, as *tokens* are masked at random, and the model is simply tasked with reconstructing them, sometimes several times for the same sentence masked at different positions.

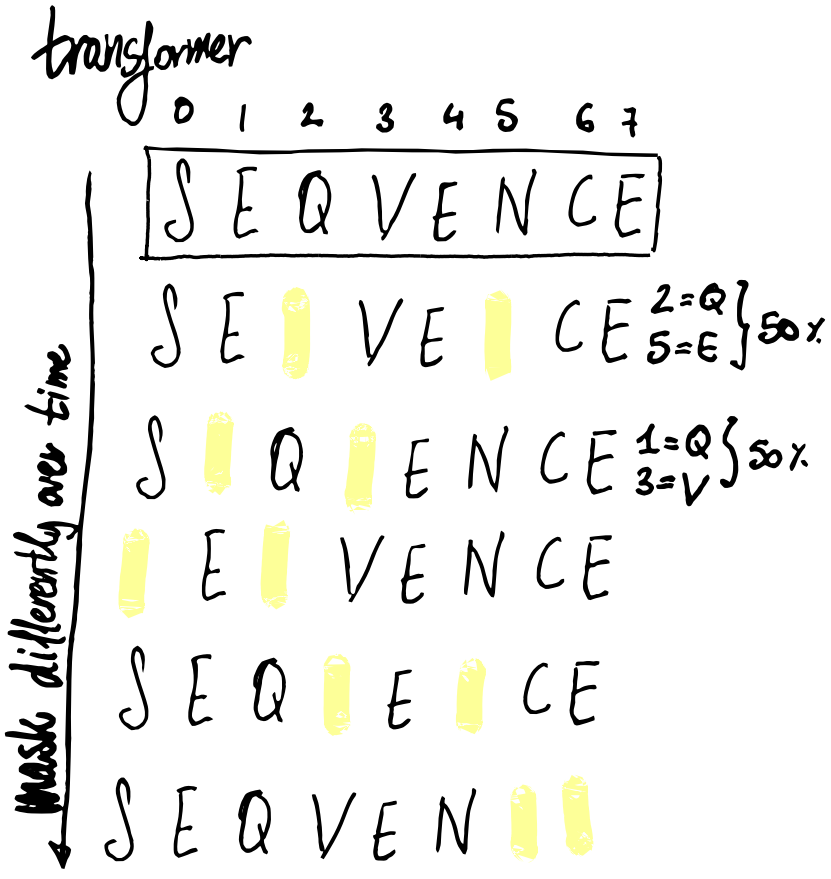


Fig. 5 – Masked Language Modelling learning task. *A transformer model learning through Masked Language Modelling (MLM) will learn how to reconstruct an input sequence (top in the black box) corrupted at random sites (yellow highlights). During training, the sequence will be corrupted at different locations increasing the ability of the model to learn the relationships between different tokens of the uncorrupted sequence. At each training step, the model is tasked with producing the most likely token at the masked positions.*

While initially pLM representations didn't quite live up to more traditional feature retrieval strategies (Alley *et al.*, 2019; Heinzinger *et al.*, 2019; Bepler and Berger, 2019), ultimately, the second variety of models both gained in speed and accuracy, topping some charts (Rives *et al.*, 2019; Elnaggar *et al.*, 2021). Nevertheless, the true advancement these models brought isn't necessarily in "beating SOTA", but in providing a different encoding of protein sequences than what either traditional methods or sequence alone could.

About two years into the development of the first pLMs (Bepler and Berger, 2019; Rao *et al.*, 2019; Alley *et al.*, 2019; Heinzinger *et al.*, 2019), two more factors influenced the scientific community towards continuing to dissect these models. First, the realization that the learned protein representations showed good performance on function tasks, and not quite on the same proteins that MSAs would (Littmann, Heinzinger, Dallago, Olenyi, *et al.*, 2021). This was a great indication that a different representation than what was available through direct evolutionary information was needed to answer some questions in bioinformatics, and maybe pLMs could enable that. Additionally, by providing continuous representations of proteins, functional predictions weren't constrained by categories, but could be assigned on a continuum (more on this in "*Conclusion*"). Second, once abstract computational representations of proteins from their sequences were obtained, the machine could be poked to go the other way around and suggest sequences to answer questions on function or structure (Madani *et al.*, 2021). Clearly, this was not a novel idea, with many applications of machine learning in *protein design* (Yang *et al.*, 2019; Ogden *et al.*, 2019; Bryant *et al.*, 2021; Frazer *et al.*, 2021). The striking difference for pLMs over other approaches is that these systematic approaches like *SeqVec* (Heinzinger *et al.*, 2019) or *ProtTrans* (Elnaggar *et al.*, 2021), which learned on *all* protein sequences available in some database, may contain a more fundamental signals, and thus encode a more overarching representation of proteins. In other words, while EVE (Frazer *et al.*, 2021) may cover well the depth of one protein, ProtTrans models may cover the width of all proteins.

2.2.2 ACCESSIBLE AND EXPLORATIVE PROTEIN MACHINE LEARNING

Accessible machine learning is a multifaceted topic, spanning from how models are made available to the community, to how predictions are consumed and interpreted by biologists. In the case of protein predictions, tools like PredictProtein (Bernhofer *et al.*, 2021), CellMap (Dallago *et al.*, 2018, 2020) or EVcouplings (Thomas A Hopf *et al.*, 2019) primarily aim at making advances in machine learning accessible and interpretable to the community by wrapping advanced research outputs into easily usable UIs with digestible visualizations. These software solutions could be built around live predictions models (e.g., PredictProtein (Bernhofer *et al.*, 2021) and EVcouplings (Thomas A. Hopf *et al.*, 2019)), or repackage predictions from an array of models into custom visualizations (e.g., CellMap (Dallago *et al.*, 2018, 2020)).

One of the major challenges in developing these solutions is to balance the ease of use and interpretability with computational overhead stemming from computing predictions or visualizations (Bernhofer *et al.*, 2021). For instance, predictions from pLMs wrapped in a webserver through bio-embeddings (Dallago, Schütze, *et al.*, 2021) required a complex, distributed setup, with pLMs running on GPU equipped machines, and feature models (like subcellular localization prediction (Stärk *et al.*, 2021)) running on CPU equipped machines. Through this setup, predictions for single sequences could be computed almost instantaneously, allowing downstream analysis on the fly. However, this setup cannot instantly predict at proteome scale, and thus different approaches, such as pre-computing predictions for an organism, may be more sensible for system-wide analyses.

Furthermore, developing sensible visualizations is another challenging task. On the one hand, visualizations help consumers grasp a view of complex topics, but conversely, they could constrain the space of exploration, or even unintentionally mislead (Kelleher and Wagener, 2011). In the case of subcellular location prediction, instantaneous predictions using pLMs coupled with cell maps (Dallago *et al.*, 2018) can highlight where a single protein may be located in a cell, but arguably it may be more beneficial to visualize this in the context of all proteins of that organism to hypothesize which proteins may interact based on their spatial proximity. Further enhancing this visualization by contextualizing proteins with their experimentally known interaction partners (i.e., overlaying protein-protein interaction information) could render more powerful hypothesis generation tools. However, as protein subcellular location prediction tools don't have tissue-specific resolution (e.g., a protein might locate in the cytoplasm in liver tissue, but in the nucleus in brain tissue), a visualization suggesting that two proteins are in different cellular compartment and interact based on experimental data might suggest a more complex mechanism than occurs.

Ultimately, how experimental and predicted biological data is presented has a direct impact on the biological interpretations that can be extracted from said data, and potentially influence discoveries (Chari *et al.*, 2021). Nevertheless, making prediction tools available for the community to use is essential to enable further discoveries, yet designing them to be effective is a big challenge (Gardner *et al.*, 2022).

3 Scientific contributions

The following sections list four publications accompanied by a short summary and relevance to this dissertation. A longer discussion about how these publications fit into the discourse outlined in the introduction will follow in the *Conclusion*. First authors are highlighted by underline. Due to *regulatory* constraints, many potentially interesting scientific contributions are added here *in spirt* only, for instance in the last section of this chapter.

3.1 CellMap visualizes protein-protein interactions and subcellular localization

Original publication. This chapter was originally published as a peer-reviewed journal article:

Dallago C, Goldberg T, Andrade-Navarro MA *et al.* CellMap visualizes protein-protein interactions and subcellular localization [version 2; peer review: 2 approved]. *F1000Research* 2018, **6**:1824 (https://doi.org/10.12688/f1000research.12707.2)

Summary. Several alternatives to visualizing protein-protein interactions (PPIs) exist. Most focus on visualizing PPIs in arbitrary spaces (e.g., two or more nodes in a network). Similarly, several tools visualizing where proteins locate in the cell exist, but these often focus on visualizing individual proteins by highlighting the areas on some predefined cell image of where proteins may appear. The novelty introduced by CellMap was to combine the two biological dimensions of protein location and interaction into a single visualization. Through the tool, an instance of which is available at cellmap.protein.properties, users can search for proteins using their UniProt (The UniProt Consortium, 2021) identifier. Once a protein of interest is identified, a protein-centric page displays information about interaction partners and localizations. From the protein-centric page a "map" view displaying the protein and its interaction partners can be opened. In this view, the cell is like a city map on google maps, while proteins are dots on the map localizing the protein in one of its possible subcellular locations. In this view, interactions between all proteins on display can be overlayed, as well as interactions of selected proteins with their partners. The tool was designed with flexibility in mind, allowing users to upload their own datasets and their own cell images, which could then be annotated with areas corresponding to localizations found in the data. By default, the tool shipped with a cartoon of a cell annotated with 13 subcellular locations, as well

as annotated data for human proteins both for their localization in multiple classes (either from experimentally annotated or predicted sources), as well as protein-protein interactions from an openly available dataset (which annotates both physical transient PPIs, as well as protein interactions by association, e.g., when two proteins are part of the same complex, although they may not physically interact).

Relevance. The seed that lead to this project was the desire to make predictions of protein properties more accessible to non-experts in an exploratory way. By jointly visualizing two protein attributes, qualitative analyses of protein interaction and hypothesis generation could be performed visually. While potentially constraining users by encoding for several biological dimensions at the same time, the biased visualization also encodes for richer context. However, users are allowed to freely modify every aspect of the visualization. For instance, while the default hosted site features location and interaction data from human, whose cells are organized into different organelles than plants, users interested in plant data and cells could upload their own data and images to be visualized.

Contribution. I am first and corresponding author of this paper. I was responsible for writing, implementation, and experimental setup.

3.2 LEARNED EMBEDDINGS FROM DEEP LEARNING TO VISUALIZE AND PREDICT PROTEIN SETS

Original publication. This chapter was originally published as a peer-reviewed journal article:

Dallago, C., Schütze, K., Heinzinger, M., Olenyi, T., Littmann, M., Lu, A. X., Yang, K. K., Min, S., Yoon, S., Morton, J. T., & Rost, B. (2021). Learned embeddings from deep learning to visualize and predict protein sets. *Current Protocols*, 1, e113. doi: 10.1002/cpz1.113

Summary. The ability of protein-based machine learning models to encode descriptive computational representations of proteins is increasingly leveraged to guide experimental decision making. Fast models that allow to classify custom sequence datasets are desired, for instance to focus experiments on more promising biotherapeutic candidates. Recently, Language Models (LMs) have been adapted from use in natural language processing (NLP) to work with

protein sequences instead. Protein LMs show enormous potential in generating descriptive representations for proteins from just their sequences at a fraction of the time compared to previous approaches. pLMs convert amino acid sequences into embeddings (vector representations) that can be used for analytical purposes, and in unsupervised and supervised pipelines for prediction of function and structure. Access to protein LMs is scattered throughout the web, a limiting factor to their use. Differently from previous approaches, any one pLM may uniquely shine light on a subset of the sequence space depending on its training objective and datasets. The bio-embeddings suite offers a unified interface to pLMs to embed large protein sets simply and quickly, to project the embeddings in lower dimensional spaces, to visualize proteins on interactive scatter plots, and to extract annotations using either supervised models, or unsupervised techniques. The array of tools offered through bio-embeddings enables quick hypothesis generation and testing and refined model optimizations on promising prediction candidates. Bio-embeddings features a pipeline which is accompanied by a web server that offers to embed, project, visualize, and extract annotations for small protein datasets directly online, without the need to install software.

Relevance. This software suite was developed as a segue to the training of protein language models (pLMs), which could compute embeddings (representations) for residues in protein sequences, which can later be used for predictions. pLM embeddings offer to shine light on proteins using representations unbiased by supervised properties, as the losses are often self-supervised (i.e.: reconstructing the syntax of proteins), although sometimes tuned on secondary losses, like encoding explicitly for structure. A solid software solution around these tools enabled quicker use of complex machine learning models for non-experts and introduced standardization. In fact, the solution presented here (bio-embeddings) could be run freely on cloud infrastructure (e.g., Google Colab), enabling researchers without cluster access of significant compute resources to use pLMs for their research, democratizing the use of cutting-edge research. Additionally, some of the pLMs and prediction methods were included in a modular web service, allowing to compute embeddings and predictions programmatically or through web UIs (e.g., embed.protein.properties or predictprotein.org) instantaneously, without local compute overhead. On top, bio-embeddings provided a one stop to find and discuss relevant research on pLMs from an array of research groups. pLM embeddings

Contribution. I am one of three principal authors. I am also corresponding author. I contributed at all stages as lead scientist.

3.3 PredictProtein – Predicting Protein Structure and Function for 29 Years

Original publication. This chapter was originally published as a peer-reviewed journal article:

Michael Bernhofer, Christian Dallago, Tim Karl, Venkata Satagopam, Michael Heinzinger, Maria Littmann, Tobias Olenyi, Jiajun Qiu, Konstantin Schütze, Guy Yachdav, Haim Ashkenazy, Nir Ben-Tal, Yana Bromberg, Tatyana Goldberg, Laszlo Kajan, Sean O'Donoghue, Chris Sander, Andrea Schafferhans, Avner Schlessinger, Gerrit Vriend, Milot Mirdita, Piotr Gawron, Wei Gu, Yohan Jarosz, Christophe Trefois, Martin Steinegger, Reinhard Schneider, Burkhard Rost, PredictProtein - Predicting Protein Structure and Function for 29 Years, *Nucleic Acids Research*, 2021;, gkab354, https://doi.org/10.1093/nar/gkab354

Summary. Since its 1992 launch, PredictProtein (https://predictprotein.org/) has been a one-stop online resource for protein analysis. In 2020, for an average of 3000 monthly users, PredictProtein combined over 13 tools into a single resource. From just an input protein sequence, the server provides online visualizations of multiple sequence alignments (MSAs), predictions of protein structure (secondary structure, solvent accessibility, transmembrane segments, disordered regions, protein flexibility, and disulfide bridges) and function (variant effect, GO terms, subcellular localization, and protein-, RNA-, and DNA binding sites). By additionally providing computable artifacts (via programmatic access), the server caters the needs of computational and experimental biologists alike. Offline use of PredictProtein tools is enabled via an omni-docker container: quickly installed on single machines and clusters. Since the previous major update in 2014, PredictProtein's infrastructure was enhanced to offer more reliable execution, more storage space and decreased runtime for predictions. Runtime was also cut four-fold by sourcing alignment generation to MMseqs2 (M Mirdita *et al.*, 2021). Usability was improved via new UI elements (Watkins *et al.*, 2017). Prediction methods for DNA-, RNA- and protein binding and GO annotations have been replaced with revised methods (Qiu *et al.*, 2020; Littmann, Heinzinger, Dallago, Olenyi, *et al.*, 2021). ProtT5-sec, an alternative secondary structure prediction method based on cutting-edge Deep Learning techniques (Elnaggar *et al.*, 2021), was integrated side-by-side to evolution-based RePROF. The PredictProtein server offers access to a vast range of accurate predictors, many topping the leaderboards even after a decade, with new recently integrated methods to boost the breadth of available sequence features and improve accuracy on dated methods.

Relevance. PredictProtein has served users with predictions of protein properties for almost 30 years. Its relevance to the field and to this thesis are manyfold: from providing landmark solutions to characterize proteins, pushing the boundaries of "known" sequence space, to integrating

intuitive visualizations to simplify interpretation of complex machine learning predictions for non-experts. The most significant scientific update in the 2021 edition of PredictProtein was the integration of cutting edge pLM models, on the one hand cementing the foundation to their use for the broader community, on the other hand, signaling a shift in how computational predictions of proteins are used. While previous models focused mainly on predicting attributes of proteins, e.g., subcellular localization, through embeddings computational biologists can access the underlying representation of proteins, enabling custom analyses of proteins from a high dimensional embedding without categorization into narrow, supervised ontologies.

Contribution. I am one of four principal authors of this paper. I am also the corresponding author. I contributed conceptualization and writing.

3.4 FLIP: BENCHMARK TASKS IN FITNESS LANDSCAPE INFERENCE FOR PROTEINS

Original publication. This chapter was originally published as a peer-reviewed conference article:

Christian Dallago, Jody Mou, Kadina E. Johnston, Bruce Wittmann, Nick Bhattacharya, Ali Madani, Kevin K. Yang, FLIP: Benchmark tasks in fitness landscape inference for proteins, *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021; https://openreview.net/forum?id=p2dMLEwL8tF

Summary. While representation learning approaches like pLMs could unlock protein design applications, no benchmark assessing their native ability to do so existed. Developing sets probing the ability of machine learning to design proteins is challenging, as some proteins are multi-purpose molecules, and current biological experiments often focus on single aspects of selected proteins. While systematic approaches like CASP and CAFA, assessing structure and function systematically respectively exist, they do not target metrics relevant for protein engineering. Fitness Landscape Inference for Proteins (FLIP) is a curated set of several biological experiments aimed at probing the ability of machine representations of proteins to support protein design campaigns. To achieve this, several *splits* from three experimental datasets were devised, testing the ability of protein representations to emulate typical experimental protein design settings, e.g., extrapolative (predicting the effect of multiple changes along the protein sequence by knowing the effect of few changes) and low-resource (predicting landscapes from only a few labelled

samples). The landscape splits come with data standardization, enabling quick adoption in computational pipelines, and enabling easy probing for new representation models.

Relevance. While probing pLMs' ability to characterize protein sequences by predicting on traditionally accepted tasks such as structure and well-defined aspect of function (e.g., subcellular localization (Stärk *et al.*, 2021)) may support their validity, these annotations are *sharp* cutouts of the "continuous" nature of protein function that may need to be captured to design proteins. Embeddings from pLMs encode continuous representations that could potentially correlate with the continuous nature of function. One attempt to correlate these realities is to predict mutational landscapes using embeddings (Marquet *et al.*, 2021). However, probing purely on deep mutational scanning (DMS) sets limited to mutational effects of single residue substitutions one at the time may not entirely characterize more complex mutational neighborhoods from a wildtype sequence. Experiments introducing a random number of residue substitutions offer a complementary approach to DMS sets. FLIP contributes by introducing four datasets for the assessment of protein representations to stack up to the continuous nature of protein function. Two of the three datasets focused on mutational landscapes from a wildtype sequence to mutated versions of it with up to 32 changes. The last dataset focused on protein thermal stability, characterizing the turning degree at which proteins start to denature (i.e., become ineffective).

Contribution. I am one of two principal authors of this paper. I contributed conceptualization, implementation, and writing.

Copyright notice. The original publication is available in open access at openreview.net/forum?id=p2dMLEwL8tF and as appendix to this manuscript.

3.5 ADDITIONAL PEER-REVIEWED SCIENTIFIC CONTRIBUTIONS

Manuscripts marked with * indicate (co-)first authorship.

On learning representations of proteins:

- **Modeling aspects of the language of life through transfer-learning protein sequences** (Heinzinger *et al.*, 2019)
  A protein language model (pLM) using LSTMs to represent protein sequences. The goal was to go beyond MSAs and find a universal protein representation applicable also to proteins for which MSAs fell short.
  https://doi.org/10.1186/s12859-019-3220-8

- **ProtTrans: Towards Cracking the Language of Life's Code Through Self-Supervised Deep Learning and High Performance Computing** (Elnaggar *et al.*, 2021)

Several pLMs using transformers to represent protein sequences. The goal was to improve on the previous approach using LSTMs by training on much larger scale.
https://doi.org/10.1109/TPAMI.2021.3095381

On applying learned protein representations to predict protein properties:

- *** Light Attention Predicts Protein Location from the Language of Life** (Stärk *et al.*, 2021)
  In this project, we characterized a novel machine learning method to predict the subcellular location of proteins in eukaryotic cells using protein embeddings. Furthermore, we studied the biases that standard datasets in protein predictions may have on reporting performance, introducing a strategy to build novel test sets to validate the veracity of accuracy estimates.
  https://doi.org/10.1093/bioadv/vbab035

- **Embeddings from deep learning transfer GO annotations beyond homology** (Littmann, Heinzinger, Dallago, Olenyi, *et al.*, 2021)
  Combining pLMs with a simplistic, unsupervised approach to predict similarly functioning proteins. The goal was to establish the potential of pLMs to predict protein function. Additionally, by using an unsupervised approach (embedding distance), exploration into similarly functioning proteins in high dimensions could be enabled.
  https://doi.org/10.1038/s41598-020-80786-0

- **Embeddings from protein language models predict conservation and variant effects** (Marquet *et al.*, 2021)
  Using pLMs to reconstruct deep mutational scanning (DMS) data and conservation of residues in protein sequences. The intent was to study if pLM embeddings could natively capture aspects of variation, for instance gain or loss of function. While the ability of pLMs to capture variation may be limited by inner workings of the machine learning models and noise in experimental DMS data, pLMs seem to well capture sequence conservation.
  https://doi.org/10.21203/rs.3.rs-584804/v2

- **Clustering FunFams using sequence embeddings improves EC purity** (Littmann, Bordin, *et al.*, 2021)
  Using pLMs to refine the annotations of functional families. This may be viewed as a segue to the publication exploring the use of pLM embeddings to annotate GO function in an unsupervised fashion.
  https://doi.org/10.1093/bioinformatics/btab371

- **Protein embeddings and deep learning predict binding residues for various ligand classes** (Littmann, Heinzinger, Dallago, Weissenow, *et al.*, 2021)
  Using pLMs to predict whether a protein binds other molecules.

https://doi.org/10.1101/2021.09.03.458869

On using software to help scientific dissemination and exploration:

- *** Visualizing Human Protein-Protein Interactions and Subcellular Localizations on Cell Images Through CellMap** (Dallago *et al.*, 2020)
  An update on the tool presented in a previous chapter with additional data on binary protein-protein interactions (between proteins verified to be physically interacting).
  https://doi.org/10.1002/cpbi.97

- **Capturing scientific knowledge in computable form** (Wong *et al.*, 2021)
  A software tool to annotate biological pathways aimed at scientific authors publishing manuscripts in biology outlets. The goal was to provide an interface to transfer expert knowledge from authors to machines, which then can then be leveraged to inform bioinformatics tools.
  https://doi.org/10.1101/2021.03.10.382333

- **Pathway Commons 2019 Update: integration, analysis and exploration of pathway data** (Rodchenkov *et al.*, 2020)
  A meta-database collecting biological pathway annotations from several curated sources. The goal was to facilitate finding biological information in an integrated resource.
  https://doi.org/10.1093/nar/gkz946

- **The EVcouplings Python framework for coevolutionary sequence analysis** (Thomas A Hopf *et al.*, 2019)
  A software tool to perform direct coupling analysis (DCA) in python. The goal was to provide an easy-to-use programmatic interface to a tool that improved structure predictions many folds.
  https://doi.org/10.1093/bioinformatics/bty862

4 Conclusion

While the introduction gave an ample overview about fundamentals in bioinformatics and where biases in data or experimental approaches arise, this section focuses on contextualizing the outcome of scientific contributions listed in the previous chapter to tackle some of the biases introduced prior.

Learning putative sequences may smoothen bias. bio-embeddings (Dallago, Schütze, *et al.*, 2021) is a tool that collects several pLMs (Heinzinger *et al.*, 2019; Bepler and Berger, 2019; Rives *et al.*, 2019; Lu *et al.*, 2020; Meier *et al.*, 2021; Elnaggar *et al.*, 2021) into a standardized software solution. pLMs are a recent innovation built on applying advanced NLP tools (Devlin *et al.*, 2019; Raffel *et al.*, 2020) most often to learn *general* representations of protein sequences from large protein sequence databases (The UniProt Consortium, 2021; Steinegger and Söding, 2018). These novel representations helped shine a different light on areas of the protein space which could not be targeted using previous tools (Littmann, Heinzinger, Dallago, Weissenow, *et al.*, 2021; Littmann, Heinzinger, Dallago, Olenyi, *et al.*, 2021; Littmann, Bordin, *et al.*, 2021; Marquet *et al.*, 2021; Meier *et al.*, 2021). By being trained, in some instances (Elnaggar *et al.*, 2021), on datasets with large amounts of putative sequences (Steinegger and Söding, 2018), the representations from pLMs may be adjusted to some selection biases present in more curated sequence datasets (The UniProt Consortium, 2021), or those using training objectives informed by domain-specific research (Bepler and Berger, 2019; He *et al.*, 2021; Min *et al.*, 2021). In fact, pLMs offer a unique opportunity to test this hypothesis, by training on the larges sets first, and fine-tuning on slightly smaller sets in a second step. This approach was adopted to train the best performing ProtTrans model (ProtT5) (Elnaggar *et al.*, 2021). While the native performance of ProtT5 trained on BFD (Steinegger and Söding, 2018) was good, it was only through fine-tuning on UniRef50 (Suzek *et al.*, 2015) that the model started showing improvements over competing solutions (Rives *et al.*, 2019) trained exclusively on the UniRef sets. The assessment of performance, in these instances, was conducted on downstream prediction tasks for experimentally annotated protein sequences that can be found directly or indirectly (via a family representative, i.e., a sequence which shares some sequence similarity to an annotated protein sequence) in the UniRef sets. Thus, while on the one hand improvement on predictions can be attributed to higher quality of the UniRef sequence databases with respect to BFD, on the one hand another interpretation of same might indicate that the fine-tuned representations are more biased towards the sequences for which annotations are known, or that the space of sequences for which experimental evidence is available are even *tighter* clustered cutouts of the larger sequence universe. Ultimately, assessing that models trained on a larger number of putative sequences may better capture general sequence

representations remains open to quantitative verification, but the practical availability of these models, which could be easily switched in- and out during analysis (Unsal *et al.*, 2020) lays the foundation to address this new research question.

Protein function on a continuum beyond ontologies. Supervised predictions using pLM representations could, in some cases, beat previous best performing methods for both structure (Rao *et al.*, 2021) and function tasks (Stärk *et al.*, 2021). However, the true advantage of continuous computational representations of protein sequences in a shared space obtained through embeddings from pLMs is that sequence sets can be projected into high dimensional representations where distances between proteins encode notions of *similarity* beyond naïve sequence similarity (Littmann, Heinzinger, Dallago, Olenyi, *et al.*, 2021). While non-space sharing solutions like protein-specific variational autoencoders models (Frazer *et al.*, 2021) promise to encode greater fine-grained, sequence-specific detail, space-sharing embeddings of pLMs enable characterizations of unsupervised functional protein clusters (Littmann, Bordin, *et al.*, 2021). Identifying clusters in these spaces could enable discoveries of proteins found in nature that can address therapeutic or industrial needs, or help classify functions of designed sequences (Dallago, Mou, *et al.*, 2021). As such, considering pLM representations for functional characterization of proteins enables to tackle the complexity of protein function annotations by circumventing the need the categorize function altogether. In fact, one could view protein function as a continuum and try to correlate it with the numerical representations from embeddings. Attempts to do so for specific functions showed some promise (Marquet *et al.*, 2021), but more importantly, low-resource (few sequence) scenarios clearly showed the potential of pLMs to better model the effect of variation (Dallago, Mou, *et al.*, 2021).

Challenging standard sets in bioinformatics. Machine learning solutions in bioinformatics sometimes operate on the assumption that an accepted dataset with labelled samples sets the standard in the field against which new approaches need to measure up. Maximizing metrics for standard dataset might grow better predictors for that problem, but since protein annotations are sharp categorizations, and our understanding of biology (especially function) requires a more flexibility especially as annotations change in time, what we may end up doing relying too much on immortalized standards is learning the distributions of these datasets, rather than generalizing the biological knowledge we wish to encode (Stärk *et al.*, 2021). One effective way to assess advance for protein predictions is to predict on many dimensions. This approach for instance highlighted that while pLMs initially did not beat previous models on any supervised prediction task (Heinzinger *et al.*, 2019), they evidently captured a diverse enough representation to come close for many, a reality which was difficult to achieve with previous methods (Unsal *et al.*, 2020)**.** Another effective way to address potential overfitting of models to standard datasets is to develop more challenging test sets to assess generalization of models post-training on the standards. For instance, in subcellular localization prediction, the standard set introduced by DeepLoc (Almagro Armenteros *et al.*, 2017) and used by several groups to claim advances in localization prediction is prone to give overestimates of performance due to similar class distributions in train and test sets.

By using a non-similarly distributed set of newly annotated protein sequences to test subcellular prediction methods, while not necessarily tackling the relative ranking of prediction methods, the overall accuracy of all methods dropped significantly (from around 80% to 60% in predicting one of ten subcellular localization classes for the best performing methods) (Stärk *et al.*, 2021). Alternatively, incorporating domain knowledge and different levels of redundancy reduction (Kandathil *et al.*, 2021) to ensure different degrees of generalization of the machine learning models may drive more reliable tools.

Accessible and explorative software tools for bioinformatics analysis. However far machine learning models push the understanding of biology, open, interpretable, and exploratory models are key to enable dissecting and disseminating advancements. For one, visualization tools expand on predictions by enabling biologists to formulate new scientific hypotheses (Dallago *et al.*, 2018, 2020). Furthermore, interactive and exploratory tools enabling collection of biological annotations directly at the source (i.e., by the authors that make scientific discoveries) ensure higher fidelity annotations for machine learning applications (Wong *et al.*, 2021). Finally, tools bundling machine learning applications into visualizations suites (Bernhofer *et al.*, 2021; Wong *et al.*, 2021; Thomas A Hopf *et al.*, 2019) democratize research, allowing researchers everywhere to characterize unknown proteins using cutting edge research.

Biases model machine learning predictions in protein biology. The presence of bias in annotations for protein biology is inevitable due to limits of experimental approaches and time constraints. However, even biased data can be leveraged via machine learning to discern underlying signals in biological data, aiding researchers in creating models of proteins and their functions. Predictive machine learning models often encode smoothened representations of a particular task, allowing some level of error correction from selecting data. The more principled the learning task, for instance *learning sequence syntax* from large sequence databases containing putative samples, the more probable the machine model of biology is less biased by selective constraints of experimental approaches.

REFERENCES

Akdel,M. *et al.* (2021) A structural biology community assessment of AlphaFold 2 applications. *BioRxiv Prepr. Serv. Biol.*

Alley,E.C. *et al.* (2019) Unified rational protein engineering with sequence-based deep representation learning. *Nat. Methods*, **16**, 1315–1322.

Almagro Armenteros,J.J. *et al.* (2017) DeepLoc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, **33**, 3387–3395.

AlQuraishi,M. (2021) Machine learning in protein structure prediction. *Curr. Opin. Chem. Biol.*, **65**, 1–8.

Ashburner,M. *et al.* (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.*, **25**, 25–29.

Bairoch,A. (2000) The ENZYME database in 2000. *Nucleic Acids Res.*, **28**, 304–305.

Barua,A. *et al.* (2021) Co-option of the same ancestral gene family gave rise to mammalian and reptilian toxins. *BMC Biol.*, **19**, 268.

Bengio,Y. *et al.* (2013) Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, **35**, 1798–1828.

Bepler,T. and Berger,B. (2019) Learning protein sequence embeddings using information from structure. *ArXiv190208661 Cs Q-Bio Stat*.

Berman,H.M. *et al.* (2000) The Protein Data Bank. *Nucleic Acids Res.*, **28**, 235–242.

Bernhofer,M. *et al.* (2021) PredictProtein - predicting protein structure and function for 29 years. *Nucleic Acids Res.*, **49**, W535–W540.

Bryant,D.H. *et al.* (2021) Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotechnol.*, **39**, 691–696.

Chari,T. *et al.* (2021) The Specious Art of Single-Cell Genomics. 2021.08.25.457696.

Ching,T. *et al.* (2018) Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface*, **15**, 20170387.

Cobb,M. (2017) 60 years ago, Francis Crick changed the logic of biology. *PLOS Biol.*, **15**, e2003243.

Dallago,C. *et al.* (2018) CellMap visualizes protein-protein interactions and subcellular localization. *F1000Research*, **6**.

Dallago,C., Mou,J., *et al.* (2021) FLIP: Benchmark tasks in fitness landscape inference for proteins. In, *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 2)*.

Dallago,C., Schütze,K., *et al.* (2021) Learned embeddings from deep learning to visualize and predict protein sets. *Curr. Protoc.*, **1**, e113.

Dallago,C. *et al.* (2020) Visualizing human protein-protein interactions and subcellular localizations on cell images through CellMap. *Curr. Protoc. Bioinforma.*, **69**, e97.

Demir,E. *et al.* (2010) The BioPAX community standard for pathway data sharing. *Nat. Biotechnol.*, **28**, 935–942.

Devlin,J. *et al.* (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv181004805 Cs*.

Dobson,C.M. (2003) Protein folding and misfolding. *Nature*, **426**, 884–890.

Elnaggar,A. *et al.* (2021) ProtTrans: Towards cracking the language of lifes code through self-supervised deep learning and high performance computing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1–1.

Fowler,D.M. and Fields,S. (2014) Deep mutational scanning: a new style of protein science. *Nat. Methods*, **11**, 801–807.

Frazer,J. *et al.* (2021) Disease variant prediction with deep generative models of evolutionary data. *Nature*, **599**, 91–95.

Gardner,P.P. *et al.* (2022) Sustained software development, not number of citations or journal choice, is indicative of accurate bioinformatic software. *Genome Biol.*, **23**, 56.

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

He,L. *et al.* (2021) Pre-training co-evolutionary protein representation via a pairwise masked language model.

Hecht,M. *et al.* (2015) Better prediction of functional effects for sequence variants. *BMC Genomics*, **16 Suppl 8**, S1.

Heinzinger,M. *et al.* (2019) Modeling aspects of the language of life through transfer-learning protein sequences. *BMC Bioinformatics*, **20**, 1–17.

Henderson,R. *et al.* (2020) Controlling the SARS-CoV-2 spike glycoprotein conformation. *Nat. Struct. Mol. Biol.*, **27**, 925–933.

Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci.*, **89**, 10915–10919.

Henzler-Wildman,K. and Kern,D. (2007) Dynamic personalities of proteins. *Nature*, **450**, 964–972.

Hopf,T.A. *et al.* (2017) Mutation effects predicted from sequence co-variation. *Nat. Biotechnol.*, **35**, 128–135.

Hopf,Thomas A *et al.* (2019) The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics*, **35**, 1582–1584.

Hopf,Thomas A. *et al.* (2019) The EVcouplings Python framework for coevolutionary sequence analysis. *Bioinformatics*, **35**, 1582–1584.

Hopf,T.A. *et al.* (2012) Three-Dimensional Structures of Membrane Proteins from Genomic Sequencing. *Cell*, **149**, 1607–1621.

Howorka,S. and Siwy,Z.S. (2020) Reading amino acids in a nanopore. *Nat. Biotechnol.*, **38**, 159–160.

Ingraham,J. *et al.* (2019) Generative Models for Graph-Based Protein Design. In, Wallach,H. *et al.* (eds), *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 15820–15831.

Jadot,M. *et al.* (2017) Accounting for Protein Subcellular Localization: A Compartmental Map of the Rat Liver Proteome*. *Mol. Cell. Proteomics*, **16**, 194–212.

Jassal,B. *et al.* (2020) The reactome pathway knowledgebase. *Nucleic Acids Res.*, **48**, D498–D503.

Jumper,J. *et al.* (2021) Highly accurate protein structure prediction with AlphaFold. *Nature*, **596**, 583–589.

Kabsch,W. and Sander,C. (1983) Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, **22**, 2577–2637.

Kandathil,S.M. *et al.* (2021) Ultrafast end-to-end protein structure prediction enables high-throughput exploration of uncharacterised proteins. *BioRxiv Prepr. Serv. Biol.*

Kanehisa,M. *et al.* (2021) KEGG: integrating viruses and cellular organisms. *Nucleic Acids Res.*, **49**, D545–D551.

Kelleher,C. and Wagener,T. (2011) Ten guidelines for effective data visualization in scientific publications. *Environ. Model. Softw.*, **26**, 822–827.

Lange,A. *et al.* (2007) Classical Nuclear Localization Signals: Definition, Function, and Interaction with Importin alpha,. *J. Biol. Chem.*, **282**, 5101–5105.

Littmann,M., Bordin,N., *et al.* (2021) Clustering FunFams using sequence embeddings improves EC purity. *Bioinforma. Oxf. Engl.*, **37**, 3449–3455.

Littmann,M., Heinzinger,M., Dallago,C., Olenyi,T., *et al.* (2021) Embeddings from deep learning transfer GO annotations beyond homology. *Sci. Rep.*, **11**, 1–14.

Littmann,M., Heinzinger,M., Dallago,C., Weissenow,K., *et al.* (2021) Protein embeddings and deep learning predict binding residues for various ligand classes. *BioRxiv Prepr. Serv. Biol.*

Lu,A.X. *et al.* (2020) Self-Supervised Contrastive Learning of Protein Representations By Mutual Information Maximization. *bioRxiv*, 2020.09.04.283929.

Madani,A. *et al.* (2021) Deep neural language modeling enables functional protein generation across families. *BioRxiv Prepr. Serv. Biol.*

Marot-Lassauzaie,V. *et al.* (2019) Spectrum of protein localization in proteomes captures evolutionary relation between species. *bioRxiv*, 845362.

Marquet,C. *et al.* (2021) Embeddings from protein language models predict conservation and variant effects. *Hum. Genet.*

McCallum,M. *et al.* (2020) Structure-guided covalent stabilization of coronavirus spike glycoprotein trimers in the closed conformation. *Nat. Struct. Mol. Biol.*, **27**, 942–949.

Meier,J. *et al.* (2021) Language models enable zero-shot prediction of the effects of mutations on protein function. *BioRxiv Prepr. Serv. Biol.*

Mildvan,A.S. (1997) Mechanisms of signaling and related enzymes. *Proteins Struct. Funct. Bioinforma.*, **29**, 401–416.

Min,S. *et al.* (2021) Pre-training of deep bidirectional protein sequence representations with structural information. *IEEE Access Pract. Innov. Open Solut.*, **9**, 123912–123926.

Mirdita,Milot *et al.* (2021) ColabFold - Making protein folding accessible to all. *BioRxiv Prepr. Serv. Biol.*

Mirdita,M *et al.* (2021) Fast and sensitive taxonomic assignment to metagenomic contigs. *Bioinforma. Oxf. Engl.*, **37**, 3029–3031.

Modi,V. and Dunbrack, Roland L,J. (2021) Kincore: a web resource for structural classification of protein kinases and their inhibitors. *Nucleic Acids Res.*

Morcos,F. *et al.* (2011) Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proc. Natl. Acad. Sci. U. S. A.*, **108**, E1293-301.

Nair,R. and Rost,B. (2002) Sequence conserved for subcellular localization. *Protein Sci.*, **11**, 2836–2847.

O'Donoghue,S.I. *et al.* (2021) SARS-CoV-2 structural coverage map reveals viral protein assembly, mimicry, and hijacking mechanisms. *Mol. Syst. Biol.*, **17**, e10079.

Ogden,P.J. *et al.* (2019) Comprehensive AAV capsid fitness landscape reveals a viral gene and enables machine-guided design. *Science*, **366**, 1139–1143.

Ottmann,C. *et al.* (2007) Structure of a 14-3-3 Coordinated Hexamer of the Plant Plasma Membrane H+-ATPase by Combining X-Ray Crystallography and Electron Cryomicroscopy. *Mol. Cell*, **25**, 427–440.

Pak,M.A. *et al.* (2021) Using AlphaFold to predict the impact of single mutations on protein stability and function. *BioRxiv Prepr. Serv. Biol.*

Perdigão,N. *et al.* (2015) Unexpected features of the dark proteome. *Proc. Natl. Acad. Sci.*, **112**, 15898–15903.

Qiu,J. *et al.* (2020) ProNA2020 predicts protein–DNA, protein–RNA, and protein–protein binding proteins and residues from sequence. *J. Mol. Biol.*, **432**, 2428–2443.

Raffel,C. *et al.* (2020) Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *ArXiv191010683 Cs Stat*.

Ramilowski,J.A. *et al.* (2015) A draft network of ligand–receptor-mediated multicellular signalling in human. *Nat. Commun.*, **6**, 1–12.

Rao,R. *et al.* (2019) Evaluating Protein Transfer Learning with TAPE. In, Wallach,H. *et al.* (eds), *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 9689–9701.

Rao,R. *et al.* (2021) MSA transformer. *BioRxiv Prepr. Serv. Biol.*

Reeb,J. *et al.* (2020) Variant effect predictions capture some aspects of deep mutational scanning experiments. *BMC Bioinformatics*, **21**, 107.

Rives,A. *et al.* (2019) Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, 622803.

Rodchenkov,I. *et al.* (2020) Pathway Commons 2019 Update: integration, analysis and exploration of pathway data. *Nucleic Acids Res.*, **48**, D489–D497.

Rost,B. (2002) Enzyme Function Less Conserved than Anticipated. *J. Mol. Biol.*, **318**, 595–608.

Rost,B. (1999) Twilight zone of protein sequence alignments. *Protein Eng. Des. Sel.*, **12**, 85–94.

Rost,B. and Sander,C. (1994) Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins*, **19**, 55–72.

Rost,B. and Sander,C. (1993a) Improved prediction of protein secondary structure by use of sequence profiles and neural networks. *Proc. Natl. Acad. Sci.*, **90**, 7558–7562.

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

Rost,B. and Sander,C. (1993b) Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.*, **232**, 584–599.

Samaras,P. *et al.* (2019) ProteomicsDB: a multi-omics and multi-organism resource for life science research. *Nucleic Acids Res.*, **48**, D1153–D1163.

Sander,C. and Schneider,R. (1991) Database of homology-derived protein structures and the structural meaning of sequence alignment. *Proteins Struct. Funct. Bioinforma.*, **9**, 56–68.

Schermelleh,L. *et al.* (2019) Super-resolution microscopy demystified. *Nat. Cell Biol.*, **21**, 72–84.

Song,W. *et al.* (2018) Cryo-EM structure of the SARS coronavirus spike glycoprotein in complex with its host cell receptor ACE2. *PLOS Pathog.*, **14**, e1007236.

Stärk,H. *et al.* (2021) Light Attention Predicts Protein Location from the Language of Life. *bioRxiv*, 2021.04.25.441334.

Steinegger,M. and Söding,J. (2018) Clustering huge protein sequence sets in linear time. *Nat. Commun.*, **9**, 1–8.

Suzek,B.E. *et al.* (2015) UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, **31**, 926–932.

Tettelin,H. *et al.* (2005) Genome analysis of multiple pathogenic isolates of Streptococcus agalactiae: Implications for the microbial "pan-genome". *Proc. Natl. Acad. Sci.*, **102**, 13950–13955.

The UniProt Consortium (2021) UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Res.*

Thul,P.J. *et al.* (2017) A subcellular map of the human proteome. *Science*, **356**.

Unsal,S. *et al.* (2020) Evaluation of methods for protein representation learning: A quantitative analysis. *BioRxiv Prepr. Serv. Biol.*

Watkins,X. *et al.* (2017) ProtVista: visualization of protein sequence annotations. *Bioinformatics*, **33**, 2040–2041.

Wittmann,B.J. *et al.* (2021) Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.*, **12**, 1026-1045.e7.

Wong,J.V. *et al.* (2021) Capturing scientific knowledge in computable form. *BioRxiv Prepr. Serv. Biol.*

Woods,D.B. and Vousden,K.H. (2001) Regulation of p53 Function. *Exp. Cell Res.*, **264**, 56–66.

Wu,Z. *et al.* (2021) Protein sequence design with deep generative models. *Curr. Opin. Chem. Biol.*, **65**, 18–27.

Yang,K.K. *et al.* (2019) Machine-learning-guided directed evolution for protein engineering. *Nat. Methods*, **16**, 687–694.

Yang,X. *et al.* (2021) Define protein variant functions with high-complexity mutagenesis libraries and enhanced mutation detection software ASMv1.0. *BioRxiv Prepr. Serv. Biol.*

Yip,K.M. *et al.* (2020) Atomic-resolution protein structure determination by cryo-EM. *Nature*, **587**, 157–161.

5 APPENDIX

5.1 CELLMAP VISUALIZES PROTEIN-PROTEIN INTERACTIONS AND SUBCELLULAR LOCALIZATION

Summary. Several alternatives to visualizing protein-protein interactions (PPIs) exist. Most focus on visualizing PPIs in arbitrary spaces (e.g., two or more nodes in a network). Similarly, several tools visualizing where proteins locate in the cell exist, but these often focus on visualizing individual proteins by highlighting the areas on some predefined cell image of where proteins may appear. The novelty introduced by CellMap was to combine the two biological dimensions of protein location and interaction into a single visualization. Through the tool, an instance of which is available at cellmap.protein.properties, users can search for proteins using their UniProt (The UniProt Consortium, 2021) identifier. Once a protein of interest is identified, a protein-centric page displays information about interaction partners and localizations. From the protein-centric page a "map" view displaying the protein and its interaction partners can be opened. In this view, the cell is like a city map on google maps, while proteins are dots on the map localizing the protein in one of its possible subcellular locations. In this view, interactions between all proteins on display can be overlayed, as well as interactions of selected proteins with their partners. The tool was designed with flexibility in mind, allowing users to upload their own datasets and their own cell images, which could then be annotated with areas corresponding to localizations found in the data. By default, the tool shipped with a cartoon of a cell annotated with 13 subcellular locations, as well as annotated data for human proteins both for their localization in multiple classes (either from experimentally annotated or predicted sources), as well as protein-protein interactions from an openly available dataset (which annotates both physical transient PPIs, as well as protein interactions by association, e.g., when two proteins are part of the same complex, although they may not physically interact).

Relevance. The seed that lead to this project was the desire to make predictions of protein properties more accessible to non-experts in an exploratory way. By jointly visualizing two protein attributes, qualitative analyses of protein interaction and hypothesis generation could be performed visually. While potentially constraining users by encoding for several biological dimensions at the same time, the biased visualization also encodes for richer context. However, users are allowed to freely modify every aspect of the visualization. For instance, while the default hosted site features location and interaction data from human, whose cells are organized into

different organelles than plants, users interested in plant data and cells could upload their own data and images to be visualized.

Contribution. I am first and corresponding author of this paper. I was responsible for writing, implementation, and experimental setup.

F1000Research F1000Research 2018, 6:1824 Last updated: 21 APR 2022

Check for updates

SOFTWARE TOOL ARTICLE

REVISED CellMap visualizes protein-protein interactions and subcellular localization [version 2; peer review: 2 approved]

Christian Dallago[1], Tatyana Goldberg[1,2], Miguel Angel Andrade-Navarro[3], Gregorio Alanis-Lobato[3], Burkhard Rost[1,4-6]

[1]Department of Informatics, Bioinformatics & Computational Biology , TUM (Technical University of Munich), Munich, Germany
[2]Center for Doctoral Studies in Informatics and its Applications (CeDoSIA), TUM (Technical University of Munich) Graduate School, Munich, Germany
[3]Faculty of Biology, Johannes Gutenberg University Mainz, Mainz, Germany
[4]Department of Biochemistry and Molecular Biophysics, Columbia University, New York City, NY, USA
[5]Institute for Food and Plant Sciences WZW, Freising, Germany
[6]Institute for Advanced Study (TUM-IAS), Munich, Germany

v2 **First published:** 11 Oct 2017, **6**:1824
https://doi.org/10.12688/f1000research.12707.1
Latest published: 01 Feb 2018, **6**:1824
https://doi.org/10.12688/f1000research.12707.2

Abstract
Many tools visualize protein-protein interaction (PPI) networks. The tool introduced here, CellMap, adds one crucial novelty by visualizing PPI networks in the context of subcellular localization, i.e. the location in the cell or cellular component in which a PPI happens. Users can upload images of cells and define areas of interest against which PPIs for selected proteins are displayed (by default on a cartoon of a cell). Annotations of localization are provided by the user or through our in-house database. The visualizer and server are written in JavaScript, making CellMap easy to customize and to extend by researchers and developers.

Keywords
subcellular location, biological visualization, protein-protein interaction

This article is included in the Bioinformatics gateway.

Open Peer Review

Approval Status ✓ ✓

	1	2
version 2 (revision) 01 Feb 2018	✓ view	
version 1 11 Oct 2017	? view	✓ view

1. **Sandra Orchard**, European Bioinformatics Institute (EMBL-EBI), European Molecular Biology Laboratory, Cambridge, UK
2. **Augustin Luna**, Harvard Medical School, Boston, USA

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Christian Dallago (christian.dallago@tum.de)

Author roles: Dallago C: Resources, Software, Visualization, Writing – Original Draft Preparation; **Goldberg T**: Conceptualization, Data Curation, Supervision, Validation, Visualization, Writing – Review & Editing; **Andrade-Navarro MA**: Data Curation, Writing – Review & Editing; **Alanis-Lobato G**: Data Curation, Writing – Review & Editing; **Rost B**: Conceptualization, Data Curation, Funding Acquisition, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work was supported by the German Research Foundation (DFG) and the Technical University of Munich, within the funding programme Open Access Publishing.
The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

How to cite this article: Dallago C, Goldberg T, Andrade-Navarro MA *et al.* **CellMap visualizes protein-protein interactions and subcellular localization [version 2; peer review: 2 approved]** F1000Research 2018, **6**:1824
https://doi.org/10.12688/f1000research.12707.2

First published: 11 Oct 2017, **6**:1824 https://doi.org/10.12688/f1000research.12707.1

F1000Research 2018, 6:1824 Last updated: 21 APR 2022

REVISED **Amendments from Version 1**

In the new version of the manuscript, we have highlighted the origin of the data sources for the example deployment of the portal on http://cell.dallago.us and the behavior of the visualization tool in case there are multiple protein localizations for a protein displayed in the protein-protein interaction network visualization page.

See referee reports

Introduction

Many tools visualize different aspects of protein-protein interaction (PPI) networks; the most prominent might be Cytoscape[1]. Existing visualizations of large PPI networks continue to be difficult to use. Some proteins interact with many hundreds or thousands of others. Often referred to as 'PPI hairballs', such hubs are in the way of understanding large data sets. Many ways have been proposed to resolve such hairballs through the addition of biologically meaningful dimensions such as pathways[2] or time[3].

Another dimension was first introduced a decade ago, namely the overlay of PPIs with subcellular localization[4]. Combining PPI networks with protein location provide an intuitive way of laying out PPI networks on a graphical representation of the cell, and might reduce the clutter from PPI hairballs. This decade-old solution[4] no longer copes with today's data, in terms of scalability nor of customizability and in terms of ease-of-use.

CellMap, the prototype introduced here, takes up on the idea of PPI visualization constrained by protein location, and provides a simple visual interface for users to explore protein location inside a cell. It presents this information in a graphically pleasant way and offers several customization features. The framework has been optimized to simplify future developments, such as the addition of further data dimensions (e.g. inclusion of protein trafficking). An instance of the tool with localization data from a previous publication that includes protein localizations of the human proteome[5] and PPI data from the Human Integrated Protein-Protein Interaction rEference (HIPPIE) resource[6] is available at http://cell.dallago.us.

Methods

Implementation

The CellMap prototype is an integrated portal that exposes API calls to retrieve images (representing cells) and protein information, as well as a frontend to visualize protein location and PPI data. The portal is fully written in JavaScript, namely in the JavaScript interpreter node.js (https://nodejs.org) for the backend and vanilla JavaScript for the frontend. The portal is deployed to the public through a Docker container. Docker is a technology that allows shipping of packaged services such as web applications to customers and users without the need to install dependencies other than the Docker engine (available through: https://www.docker.com). For the representation of cell images as maps, the Leaflet framework is used. Leaflet is a JavaScript-based tool used to represent maps (http://leafletjs.com).

Data about proteins are stored as JSON documents in a Mongo (http://mongodb.com) database. All information about the interaction partners and the subcellular localization of a protein is stored in a single JSON document, making the data structure simple to understand for non-experts and enabling them to deploy prototypes using their own data. Figure 1 schematically represents a protein data model (for a specific example for a protein object: http://cell.dallago.us/api/proteins/search/Q99943).

Operation

In CellMap, users can choose to upload new maps (images of cells). They can modify the location of regions of interest (ROIs) for a selected map (Figure 2), and visualize the locations of selected proteins on a map or render protein-protein interaction networks from a set of selected proteins.

To maintain a consistent coloring scheme for different cellular compartments throughout a set of different images, each compartment is assigned a unique color through the hash of the compartment's name (e.g. light blue = vacuole, Figure 3B). Using this coloring approach, users might eventually learn to associate color with compartment. When proteins are loaded into the map, they are assigned pseudo-random coordinates representing a point that lies within the boundaries of the ROI in which they are localized (Figure 3D). A circle of a given radius is placed on the randomly generated point (Figure 3E-F), and the circle will be filled with the same color as the compartment in which the protein is located in (Figure 3B and 3F).

Users can choose between two visualization options: the subcellular location in the context of the protein-protein interaction

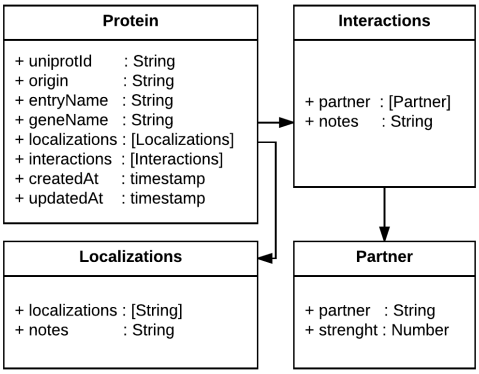


Figure 1. Diagram of the data representation in CellMap. In the figure we present a diagram of the Protein class, which contains several attributes of type *String*, two fields of type *timestamp* and two arrays (in square brackets) that reference the *Interactions* and *Localizations* classes. The arrows highlight the referenced models. This simple representation of information about a protein, its protein-protein interaction partners and its localizations enables the tool to be reused with one's own datasets.

Figure 2. Section of a screenshot of the CellMap editing tool on a private instance of the portal. In the screenshot, an authorized user with editing capabilities draws a polygon (dark green) representing a new cellular compartment or region of interest (ROI). The user has a set of tools on the left side that can be used to draw polygons, lines, squares or circles. Once the new region has been drawn, the user can associate a cellular compartment through the dropdown input on the top-right and submit the new information to the server. The image used for this screenshot was taken from Wikimedia's user Royroydeb, under CC BY-SA 4.0 (http://bit.ly/2fuYRiE) and is used in this figure for demonstrative purposes only, as using it on the online version of CellMap would infringe copyright.

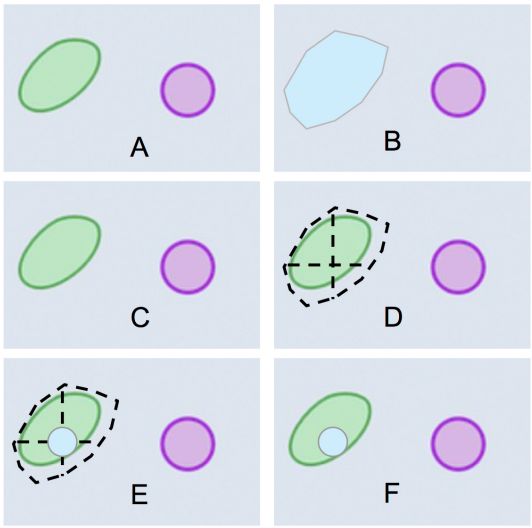


Figure 3. Definition of an area and drawing of protein circle. (**A**) Section of a cartoon image of a cell; (**B**) user-drawn polygon representing the area occupied by a vacuole; (**C**) how the section of the cartoon image is displayed on the PPI/map viewer; (**D**) random point calculation inside vacuole-polygon-defined area; (**E**) drawing of a protein circle located inside the vacuole, (**F**) result of loading a protein localized in the vacuole as shown by PPI/map viewer.

viewer (PPI viewer, Figure 4A, http://cell.dallago.us/ppi), and the protein subcellular location viewer (Map viewer, Figure 4B, http://cell.dallago.us/map). The two viewers can load the same images of cells (maps) and collect localization data from the same source, in the publicly available instance by 5. The PPI viewer offers the possibility to overlay networks between proteins being visualized. The map viewer displays all locations reported for a given protein simultaneously, while the PPI viewer only displays only one location at a time (by default: the first localization in the array of localizations as described in the protein data model, Figure 1); users can manually change the location by clicking on the protein circle and selecting a new location from the information box (Figure 5). Both the PPI and the map viewer are enriched by several controls (Figure 6): The top-left controls enable actions including: the navigation to the home of CellMap (Figure 6, panels 1 and 2, A), switching from the map viewer to the PPI viewer and vice versa, keeping the proteins currently loaded in the view (Figure 6, panels 1 and 2, B), reducing the opacity of the cell map, highlighting the protein circles (Figure 6, panels 1 and 2, C), zooming in- and out of the map and PPI viewers (Figure 6, panels 1 and 2, D), and visualization of the global network among all proteins loaded in the visualizer (Figure 6, panel 1, E). The top-right control allows to temporarily hide loaded proteins or activate an overlay of the user-drawn localizations (Figure 6, panel 4). The top-center search panel allows users to load new proteins by searching for their UniProt identifier, primary gene or primary protein name[7] into the viewer (Figure 6, panel 3).

To facilitate the retrieval of proteins and their interacting partners, CellMap provides basic search functionalities. Users can search for proteins based on their UniProt identifiers, by their gene identifiers or by their protein names. When performing the search, the page renders a grid containing boxes, each representing a different protein (Figure 7). Inside the boxes, the UniProt identifier

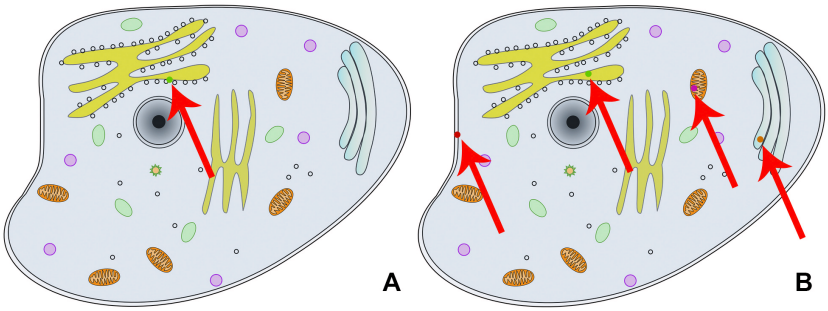


Figure 4. Comparison between PPI viewer and map viewer. The left view (**A**) shows the PPI viewer, which depicts the result of loading protein Q9NR71 and displays a circle for the first localization found in the array of locations (http://cell.dallago.us/ppi?p=Q9NR71); The right panel (**B**) shows the Map viewer, which depicts the result of loading the same protein Q9NR71 and displays a circle for the protein in each of its reported location (http://cell.dallago.us/map?p=Q9NR71). The red arrows are overlaid on top of the screenshots to highlight where the protein circles have been drawn in the viewers, since fitting the screenshot on the page reduces the overall size of the images.

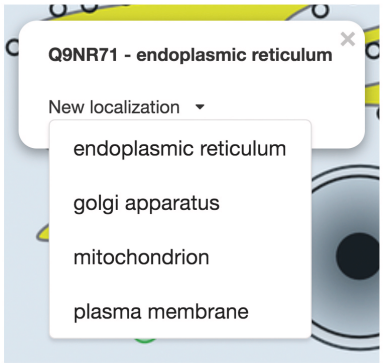


Figure 5. Protein information box. Top: information about the selected protein. Bottom: new localization selection box rendered in the PPI viewer when clicking on the protein circle (http://cell.dallago.us/ppi?p=Q9NR71).

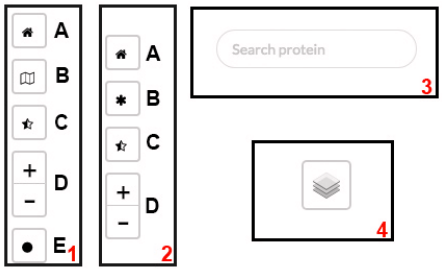


Figure 6. Controls used in the different viewers. (**1**) Top-left controls of PPI viewer; (**2**) top-left controls of map viewer; (**3**) top-center search panel of PPI/map viewer; (**4**) top-right layer control on PPI/map viewer.

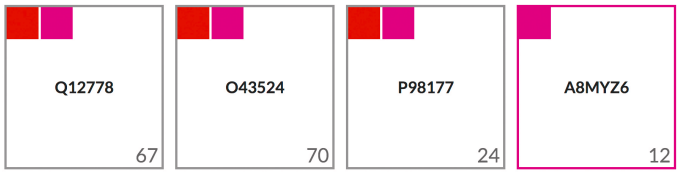


Figure 7. Results of searching for protein "*foxo*". The screenshot of this section of the home page shows four proteins that match the search criterion "*foxo*" either by their UniProt identifier, primary gene name or primary protein name. The protein boxes contain the UniProt identifiers of the matched proteins (center) and display the number of interaction partners (bottom-right) and several color-filled boxes graphically representing the localizations reported for the matched proteins (top-left).

for the protein that matched the search criterion is displayed. Starting on the top-right of every box a smaller colored square for each compartment is displayed in which that protein is localized. For proteins annotated to be in a single compartment, the border of the outer box (representing one protein as indicated by the UniProt ID in the center of the box) will get the color of that compartment (2nd box in Figure 7). Clicking on one of the colored squares will filter results based on the compartment represented by that color. In the bottom-right of each box, the total number of PPI partners are annotated.

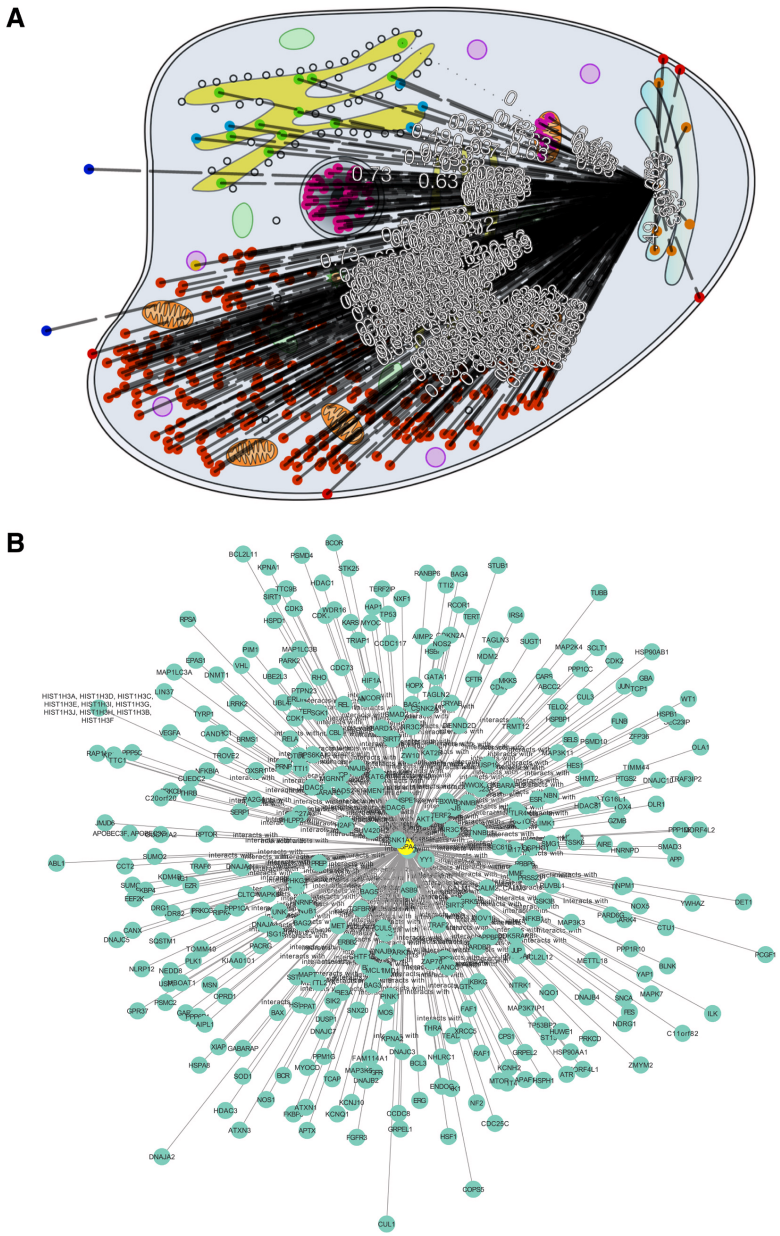
Discussion

Some CellMap functionality is exemplified by a heat shock protein (HSPA4; Heat shock 70 kDa protein 4, UniProt identifier P34932) with many interaction partners (338, according to HIPPIE, http://cbdm-01.zdv.uni-mainz.de/~mschaefer/hippie/query.php?s=HSPA4) in different compartments. The objective was to showcase how CellMap can simplify PPI hairballs. We visualize the same PPI network using CellMap (Figure 8A) and Cytoscape[1] in the form of the Cytoscape.js version used by HIPPIE (Figure 8B) and the Cytoscape desktop version (Figure 8C).

None of the three viewers solves the PPI hairball problem completely. Without zooming in, the information density for 338 protein pairs is too high to be helpful. HIPPIE's layout for Cytoscape.js (Figure 8B) clearly improves over the standard Cytoscape desktop version (Figure 8C) by centering the view around HSPA4, the protein of interest. In CellMap (Figure 8A) the biologically relevant differences between pairs from the same and from different compartments remain visible.

By using a biologically relevant dimension (protein localization), instead of drawing nodes in positions based on edge weight (force layout of Cytoscape), some aspects of the protein and its partners become obvious at first glance, e.g. that HSPA4 interacts with many nuclear and cytoplasmic proteins, as well as with proteins that are secreted (extra-cellular) and located in the Endoplasmic Reticulum (ER, Figure 8). This may suggest the hypothesis HSPA4 to be an important hub involved in process spanning across compartments. Such a hypothesis is presented in our supplementary material (Figure SOM_1), where we analyze the visualization of the FOXO3 protein through CellMap.

One disadvantage of CellMap over the Cytoscape.js view is that the protein identifiers are not visible at all on the static image (protein identifiers become visible through mouse-over events in CellMap). However, in the image shown (Figure 8) the Cytoscape.js names also remain unreadable. Another problem with CellMap are the numbers displayed on edges (experimental


A
B

F1000Research 2018, 6:1824 Last updated: 21 APR 2022

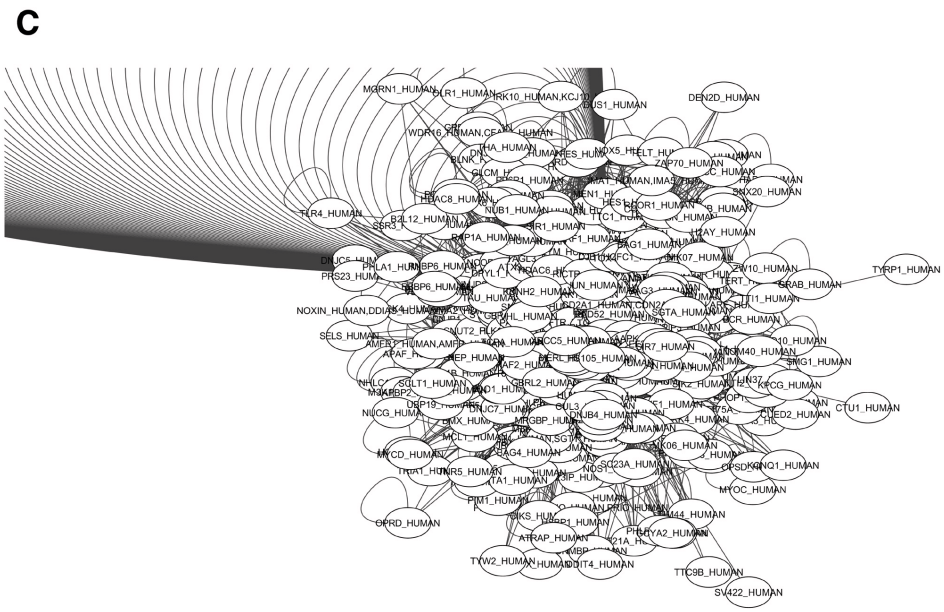


Figure 8. PPI hub in CellMap (**A**), Cytoscape.js (**B**) and Cytoscape desktop (**C**). For HSPA4 (Heat shock 70 kDa protein 4, UniProt identifier P34932), we show some of the PPIs known (according to HIPPIE HSPA4 has 338 interaction partners). We chose this as one example of a protein with many more PPIs than the average protein ("PPI hub"). The figure compares how three different PPI viewers cope with the HSPA4 network: (**A**) CellMap (http://cell.dallago.us/protein/P34932), (**B**) HIPPIE's Cytoscape.js visualizer and (**C**) the desktop version of Cytoscape. Proteins in CellMap are represented as colored dots on the map (image) of the cell, and upon selecting the protein of interest an overlay of edges is drawn. In Cytoscape and Cytoscape.js, proteins are represented as nodes containing a label (protein name as UniProt identifier), and edges are directly inferred from the data. The Cytoscape.js visualization was taken directly from HIPPIE. The Cytoscape network was automatically drawn upon loading the HIPPIE dataset and selecting the protein of interest and it's direct neighbors.

reliability of the PPI as given by HIPPIE). In our view, this information is extremely important to look at interactions, but we are still lacking a more sophisticated mechanism to visualize these numbers.

CellNetVis[8] is a recent tool that also connects localization with PPI networks. It emphasizes the way PPI networks are laid out through the adaptation of a so-called force-directed layout (using the tool While). Although CellMap and CellNetVis are founded on a similar idea, user experience and focus differ importantly. For instance, CellMap can be driven by data from users that define the number of compartments on a map, and provide localizations. In contrast, CellNetVis uses a fixed subset of compartments and an ad hoc diagram for the cell. Additionally, CellMap comes with out of the box data for the human proteome and allows the community to grow the tool by enriching datasets (images and localizations), whereas CellNetVis has a per-use approach, allowing to visualize networks stored in specialized XGMML files. Another unique aspect of CellMap is the openness to introduce further biologically meaningful dimensions (beyond location such as time or pathways) that increase the usefulness of PPI visualization tools to create new testable hypotheses.

Conclusions

CellMap is a prototype providing a portal exploring the idea of using protein subcellular location as the basis to construct more complete visualizations of biological data, such as protein-protein interactions (PPIs). Using this paradigm, we claim that additional information, such as pathways, can be layered on top of the current visualization of subcellular location to potentially generate meaningful biological insights. The source code for the portal is publicly available and an instance of the portal with location data from a previous publication about the subcellular localization of the human proteome[5] and protein-protein interaction data from

HIPPIE[6] (http://cbdm-01.zdv.uni-mainz.de/~mschaefer/hippie) is running at http://cell.dallago.us. The visualization tool is written in JavaScript, thereby tapping into a very large user base for customized extensions and modifications. With the release of the prototype, we aim at creating a user base and awareness of the tool, ultimately collecting precious feedback from experimentalists and technical users alike.

Abbreviations

2D: two dimensions, API: Application Program Interface, ID: identifier, JSON: JavaScript Object Notation, PPI: protein-protein interaction, ROI: region of interest.

Software availability

The CellMap prototype is released as open source software under the GNU General Public License v3.0. Documentation, source code and viewer are available at https://github.com/sacdallago/cellmap. Archived source code as at the time of publication is available at https://doi.org/10.5281/zenodo.904324[9]. An example of use with protein localization data from a recent publication[5] and from the HIPPIE database of protein-protein interactions[6] is available at http://cell.dallago.us.

Competing interests

No competing interests were disclosed.

Grant information

This work was supported by the German Research Foundation (DFG) and the Technical University of Munich, within the funding programme Open Access Publishing.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgements

Thanks primarily to Tim Karl, but also to Guy Yachdav (all TUM) for invaluable help with hardware and software; to Inga Weise (TUM) for support with many other aspects of this work; to Dr. Luisa Jiménez-Soto (Max von Pettenkofer-Institut) for helpful comments on the manuscript; the LRZ Compute Cloud team for hosting the webserver; to Rolf Apweiler (UniProt, EBI, Hinxton), Amos Bairoch (CALIPHO, SIB, Geneva), Ioannis Xenarios (Swiss-Prot, SIB, Geneva), and their crews for maintaining excellent databases and to all experimentalists who enabled this analysis by making their data publicly available.

References

1. Shannon P, Markiel A, Ozier O, *et al.*: **Cytoscape: a software environment for integrated models of biomolecular interaction networks.** *Genome Res.* 2003; **13**(11): 2498–2504.
PubMed Abstract | Publisher Full Text | Free Full Text

2. Chaurasia G, Malhotra S, Russ J, *et al.*: **UniHI 4: new tools for query, analysis and visualization of the human protein-protein interactome.** *Nucleic Acids Res.* 2009; **37**(Database issue): D657–D660.
PubMed Abstract | Publisher Full Text | Free Full Text

3. Ma DK, Stolte C, Krycer JR, *et al.*: **SnapShot: Insulin/IGF1 Signaling.** *Cell.* 2015; **161**(4): 948–948.e1.
PubMed Abstract | Publisher Full Text

4. Ofran Y, Yachdav G, Mozes E, *et al.*: **Create and assess protein networks through molecular characteristics of individual proteins.** *Bioinformatics.* 2006; **22**(14): e402–e407.
PubMed Abstract | Publisher Full Text

5. Ramilowski JA, Goldberg T, Harshbarger J, *et al.*: **A draft network of ligand-receptor-mediated multicellular signalling in human.** *Nat Commun.* 2015; **6**: 7866.
PubMed Abstract | Publisher Full Text | Free Full Text

6. Alanis-Lobato G, Andrade-Navarro MA, Schaefer MH: **HIPPIE v2.0: enhancing meaningfulness and reliability of protein-protein interaction networks.** *Nucleic Acids Res.* 2017; **45**(D1): D408–D414.
PubMed Abstract | Publisher Full Text | Free Full Text

7. The UniProt Consortium: **UniProt: the universal protein knowledgebase.** *Nucleic Acids Res.* 2017; **45**(D1): D158–D169.
PubMed Abstract | Publisher Full Text | Free Full Text

8. Heberle H, Carazzolle MF, Telles GP, *et al.*: **CellNetVis: a web tool for visualization of biological networks using force-directed layout constrained by cellular components.** *bioRxiv.* 2017.
Publisher Full Text

9. Dallago C: **CellMap: open software for PPI and protein localization visualization in JavaScript.** *Zenodo.* 2017.
Data Source

Open Peer Review

Current Peer Review Status: ✔ ✔

Version 2

Reviewer Report 12 February 2018

https://doi.org/10.5256/f1000research.15085.r30439

✔ **Sandra Orchard**
European Bioinformatics Institute (EMBL-EBI), European Molecular Biology Laboratory, Cambridge, UK

The authors have addressed my concerns and I have no further comments to add.

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Version 1

Reviewer Report 18 January 2018

https://doi.org/10.5256/f1000research.13762.r29754

✔ **Augustin Luna**
Dana-Farber Cancer Institute, Harvard Medical School, Boston, MA, USA

The tool provides an interesting feature to help declutter visualizations of biological networks using localization information. Some comments:

- It would be good if the names of the used databases was stated in the last paragraph of the introduction.

- The tool would be more intuitive for new users, if it provided descriptions the various colors used on the site with the same explanation as in the paper. For example, the colored boxes that represent localizations in the search results and the dot colors used for the protein visualization on the cell map.
- It is unclear from the paper all the types of interactions might be shown in the represented networks.
- Also, it is unclear from the paper, what happens to the network visualization in the cases where the identified proteins are present in multiple locations.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Yes

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 29 Jan 2018

Christian Dallago, TUM (Technical University of Munich), Munich, Germany

Dear Dr. Luna,

Thank you very much for your input on our work.

We have submitted a new version of the manuscript, which should address points one and four of your comment.

As to the second point: we have created a new feature item for our next release that displays a button on the map viewer to display a modal with the legend. As of now: a legend

is available by scrolling down to the second half of the page in the map or ppi viewers (e.g. http://cell.dallago.us/map/573c87c182a9e1ae1e37d08e?p=P04637) and expanding the "Legend" tab. We understand that this can be overseen and improved, therefore we thank you for the input.

As to the third point: in this manuscript, we focus on discussing the software implementation and visualization abilities of CellMap, rather than the data sources used in the example deployment hosted on http://cell.dallago.us. More information about the types of interactions reported by the HIPPIE data source can be found in the latest paper describing HIPPIE (http://nar.oxfordjournals.org/content/early/2016/10/28/nar.gkw985) and directly on the HIPPIE information page (http://cbdm-01.zdv.uni-mainz.de/~mschaefer/hippie/information.php#sources).

Please, feel free to suggest any other changes to both our manuscript and tool.

Best regards,
Christian Dallago.

Competing Interests: No competing interests were disclosed.

Reviewer Report 13 November 2017

https://doi.org/10.5256/f1000research.13762.r27544

?

Sandra Orchard
European Bioinformatics Institute (EMBL-EBI), European Molecular Biology Laboratory, Cambridge, UK

The authors describe a tool for visualising PPI networks in the context of the subcellular localisation of the searched protein.

1. I have twice tried to review this paper and both times the http://cell.dallago.us link gave me a MongoDB error. I have therefore had to review the paper without being able to view the tool. This is not satisfactory. I was unable to test the conclusions about the tool and its findings.

2. The tool uses a static interaction compilation database (HIPPIE) as the source of PPIs. Did the authors not consider using the PSICQUIC web service, which gives the users considerably more options as to where to source their PPI data from, and also allows the visualisation of protein-small molecule interactions and also potentially the site of action of protein-drug interactions, also available via PSICQUIC. It would also allow the data to be as up to date as the latest release of each database, which will be more frequent than releases

of HIPPIE.

3. I am not clear where the subcellular location data comes from. This may be obvious to regular users of CellMap but not to me, an should be stated in the paper for other user who do not know this.

Is the rationale for developing the new software tool clearly explained?
Yes

Is the description of the software tool technically sound?
Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?
Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?
Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?
Partly

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 25 Nov 2017

Christian Dallago, TUM (Technical University of Munich), Munich, Germany

Dear Dr. Orchard,

Thank you for your valuable input on the tool!

With regard to point 1: we apologize for the broken database connection, unfortunately, the deployment system missed that flag and thus didn't restart the service. We have fixed the issue and the website is now running. Up until now, I have not identified any other issues that could prevent the web server to run properly.

With regard to point 2: the software presented in this paper has a dual-purpose. On the one hand, we want to give the ability to discover protein localization and protein-protein

interaction from two known sources (HIPPIE for PPI, and subcellular localization from a publication, which describes localization for the human proteome based on a consensus of experimental data and state-of-the-art prediction models (http://doi.org/10.1038/ncomms8866)). On the other hand, we want to propose a system that can be reused on user-defined data (as long as it complies with the format the visualization tool digest, as from Figure 1) and be integrated as JavaScript visualization tool in different portals. For now, we would like to avoid having a direct integration of the portal with external tools via, for example, API calls. In an upcoming version of the portal, we will offer scripts to populate the database from different sources for the two data entities (protein localization and interaction).

PSICQUIC generates interaction data on-demand, which can later be downloaded. Obtaining the data requires some time: a user input one specific protein identifier, selects the databases to use to collect interaction data, submits a cluster job and finally gets access to the data. Searching for protein P45381 identified 80 interactions in all online databases. After several hours, the job was not finished, so we decided to lower the number of databases to fetch information from. Reducing the number of databases produced results quickly. The results page of PSICQUIC presents a table of interactions and visualizes a graph, which we could not load due to lack of compatibility with the Chrome browser. We believe it would be interesting to present CellMap at the level of this resource and will contact the authors of the tool to discuss what the best idea in this regard would be.
Fetching the data from PSICQUIC as it is now and putting it into the portal requires to also normalize the PSICQUIC data and map it to protein localization data. Writing a parser for the PSI-MITAB tables is straightforward, the normalization and mapping of identifiers should occur externally to CellMap. We will create a guide on how this can be done in the next days and put it on the landing page of CellMap.
Integrating protein-molecule data and displaying these entities meaningfully is an interesting idea for the future development of the CellMap tool.

With regard to point 3: the data about protein localization stems from a publication of our group (http://doi.org/10.1038/ncomms8866). The data on protein subcellular localization for humans published through this paper was the starting point for the development of CellMap. In the current manuscript, we focused more on describing the visualization tool, rather than going into detail about how the localization data was retrieved (which in this case is by building a consensus over experimental (where available) and predicted localisations for 6 subcellular compartments). This is again because we didn't want to develop a tool around this specific data source, but rather offer the possibility to change the origin for the localization data in the future.

We appreciate the suggestions for further data sources and data entities that can be used and integrated into CellMap. In upcoming releases, we will make sure to offer a bigger variety of data sources and scripts to populate and update the information on protein subcellular localization, and protein-protein interaction data used by the visualization tool. Additionally, we will contact the authors of PSICQUIC to discuss if it would be possible to integrate CellMap in the results page of a cluster job.

Best regards,
Christian Dallago, Tatyana Goldberg & Burkhard Rost.

Competing Interests: No competing interests were disclosed.

5.2 Learned Embeddings from Deep Learning to Visualize and Predict Protein Sets

Summary. The ability of protein-based machine learning models to encode descriptive computational representations of proteins is increasingly leveraged to guide experimental decision making. Fast models that allow to classify custom sequence datasets are desired, for instance to focus experiments on more promising biotherapeutic candidates. Recently, Language Models (LMs) have been adapted from use in natural language processing (NLP) to work with protein sequences instead. Protein LMs show enormous potential in generating descriptive representations for proteins from just their sequences at a fraction of the time compared to previous approaches. pLMs convert amino acid sequences into embeddings (vector representations) that can be used for analytical purposes, and in unsupervised and supervised pipelines for prediction of function and structure. Access to protein LMs is scattered throughout the web, a limiting factor to their use. Differently from previous approaches, any one pLM may uniquely shine light on a subset of the sequence space depending on its training objective and datasets. The bio-embeddings suite offers a unified interface to pLMs to embed large protein sets simply and quickly, to project the embeddings in lower dimensional spaces, to visualize proteins on interactive scatter plots, and to extract annotations using either supervised models, or unsupervised techniques. The array of tools offered through bio-embeddings enables quick hypothesis generation and testing and refined model optimizations on promising prediction candidates. Bio-embeddings features a pipeline which is accompanied by a web server that offers to embed, project, visualize, and extract annotations for small protein datasets directly online, without the need to install software.

Relevance. This software suite was developed as a segue to the training of protein language models (pLMs), which could compute embeddings (representations) for residues in protein sequences, which can later be used for predictions. pLM embeddings offer to shine light on proteins using representations unbiased by supervised properties, as the losses are often self-supervised (i.e.: reconstructing the syntax of proteins), although sometimes tuned on secondary losses, like encoding explicitly for structure. A solid software solution around these tools enabled quicker use of complex machine learning models for non-experts and introduced standardization. In fact, the solution presented here (bio-embeddings) could be run freely on cloud infrastructure (e.g., Google Colab), enabling researchers without cluster access of significant compute resources to use pLMs for their research, democratizing the use of cutting-edge research. Additionally, some of the pLMs and prediction methods were included in a modular web service, allowing to compute embeddings and predictions programmatically or through web UIs (e.g., embed.protein.properties or predictprotein.org) instantaneously, without local compute

overhead. On top, bio-embeddings provided a one stop to find and discuss relevant research on pLMs from an array of research groups. pLM embeddings

Contribution. I am one of three principal authors. I am also corresponding author. I contributed at all stages as lead scientist.

Learned Embeddings from Deep Learning to Visualize and Predict Protein Sets

Christian Dallago,[1,2,12,13] Konstantin Schütze,[1,12] Michael Heinzinger,[1,2,12] Tobias Olenyi,[1] Maria Littmann,[1,2] Amy X. Lu,[3] Kevin K. Yang,[4] Seonwoo Min,[5] Sungroh Yoon,[5,6] James T. Morton,[7] and Burkhard Rost[1,8,9,10,11]

[1]TUM (Technical University of Munich) Department of Informatics, Bioinformatics & Computational Biology, Garching/Munich, Germany
[2]TUM Graduate School, Center of Doctoral Studies in Informatics and its Applications (CeDoSIA), Garching/Munich, Germany
[3]Department of Computer Science, University of Toronto, Toronto, Canada & Vector Institute
[4]Microsoft Research New England, Cambridge, Massachusetts
[5]Department of Electrical and Computer Engineering, Seoul National University, Seoul, South Korea
[6]Interdisciplinary Program in Bioinformatics, Seoul National University, Seoul, South Korea
[7]Center for Computational Biology, Flatiron Institute, New York, New York
[8]Institute for Advanced Study (TUM-IAS), Garching/Munich, Germany
[9]TUM School of Life Sciences Weihenstephan (WZW), Freising, Germany
[10]Columbia UniversityDepartment of Biochemistry and Molecular Biophysics, New York, New York
[11]New York Consortium on Membrane Protein Structure (NYCOMPS), New York, New York
[12]These authors contributed equally to this work.
[13]Corresponding author: *christian.dallago@tum.de*

Models from machine learning (ML) or artificial intelligence (AI) increasingly assist in guiding experimental design and decision making in molecular biology and medicine. Recently, Language Models (LMs) have been adapted from Natural Language Processing (NLP) to encode the implicit language written in protein sequences. Protein LMs show enormous potential in generating descriptive representations (embeddings) for proteins from just their sequences, in a fraction of the time with respect to previous approaches, yet with comparable or improved predictive ability. Researchers have trained a variety of protein LMs that are likely to illuminate different angles of the protein language. By leveraging the *bio_embeddings* pipeline and modules, simple and reproducible workflows can be laid out to generate protein embeddings and rich visualizations. Embeddings can then be leveraged as input features through machine learning libraries to develop methods predicting particular aspects of protein function and structure. Beyond the workflows included here, embeddings have been leveraged as proxies to traditional homology-based inference and even to align similar protein sequences. A wealth of possibilities remain for researchers to harness through the tools provided in the following protocols. © 2021 The Authors. Current Protocols published by Wiley Periodicals LLC.

CURRENT PROTOCOLS
A Wiley Brand

Current Protocols e113, Volume 1
Published in Wiley Online Library (wileyonlinelibrary.com).
doi: 10.1002/cpz1.113

The following protocols are included in this manuscript:

Basic Protocol 1: Generic use of the *bio_embeddings* pipeline to plot protein sequences and annotations
Basic Protocol 2: Generate embeddings from protein sequences using the *bio_embeddings* pipeline
Basic Protocol 3: Overlay sequence annotations onto a protein space visualization
Basic Protocol 4: Train a machine learning classifier on protein embeddings
Alternate Protocol 1: Generate 3D instead of 2D visualizations
Alternate Protocol 2: Visualize protein solubility instead of protein subcellular localization
Support Protocol: Join embedding generation and sequence space visualization in a pipeline

Keywords: deep learning embeddings • machine learning • protein annotation pipeline • protein representations • protein visualization

How to cite this article:
Dallago, C., Schütze, K., Heinzinger, M., Olenyi, T., Littmann, M., Lu, A. X., Yang, K. K., Min, S., Yoon, S., Morton, J. T., & Rost, B. (2021). Learned embeddings from deep learning to visualize and predict protein sets. *Current Protocols*, *1,* e113.
doi: 10.1002/cpz1.113

INTRODUCTION

Protein sequences correspond to strings of characters, each representing an amino acid (referred to as residues when joined in a protein). While protein savants extrapolate a wealth of information from this representation, for machines this is as meaningless as any other text document. Finding meaningful, computable representations from protein sequences by converting text into vectors of numbers representing relevant features or descriptors of proteins is an important first step to find out properties of the protein with that sequence, e.g., what other proteins it resembles (sequence comparisons through alignments), what it looks like (membrane or water-soluble, regular globular or disordered), or what it does (enzyme or not, process involved in, molecular function, interaction partners).

Many approaches to generate knowledge and meaning from protein sequences have been proposed. Intuitive representations relied on what experts considered informative, e.g., converting sequences into numerical vectors representing polarity or hydrophobicity. More advanced ideas included substitution matrices (Henikoff & Henikoff, 1992), profiles of protein families (Stormo, Schneider, Gold, & Ehrenfeucht, 1982), and "evolutionary couplings" from events correlating the mutability at two or more residues (Morcos et al., 2011). Combining "evolutionary information" (Rost & Sander, 1993), along with global (entire protein) and local (only sequence fragment) features through machine learning (ML; Rost & Sander, 1993, 1994), led to the first breakthrough in protein structure prediction over two decades ago (Moult, Pedersen, Judson, & Fidelis, 1995; Rost & Sander, 1995). Combining more sophisticated tools from Artificial Intelligence (AI) to include even more protein evolutionary information have led to the most recent breakthrough by AlphaFold2 from DeepMind (Callaway, 2020).

Representations based on evolutionary information have improved remote homology detection (Steinegger et al., 2019) as well as the prediction of aspects of protein structure (Hopf et al., 2012; Rost, 1996) and protein function (Goldberg et al., 2014; Hopf et al., 2017). The amount of evolutionary information contained in these representations is proportional to the size and diversity of a protein family (Ovchinnikov et al., 2017; Rost, 2001); the generation of families relies on parameter-sensitive multiple sequence alignments (MSAs) that, due to growing databases, become increasingly computationally expensive, despite immense advances in method development (Steinegger & Söding, 2018; Steinegger, Mirdita, & Söding, 2019).

Deep learning−based Language Models (LMs) are a new class of machine learning devices learning the rules for semantics and syntax directly and autonomously from the statistics of text corpora. Modern LMs learn to represent language by being conditioned on either predicting the next word in a sentence given previous context, or by reconstructing corrupted text. In protein bioinformatics, these devices are trained on large sequence datasets, such as UniProt (The UniProt Consortium, 2019), through a process called "self-supervision". LM representations (embeddings) have been used as input to other methods (a process referred to as transfer learning) to predict aspects of protein structure and function. Although embedding-based predictions tend to be less accurate than those using evolutionary information, they require less time (Heinzinger et al., 2019; Rao et al., 2019; Rives et al., 2019). By learning to represent sequence and background information, embeddings open the door to a completely new way of using protein sequences, successful enough to even compete with traditional remote homology detection and structural alignments (Biasini et al., 2014; Littmann, Heinzinger, Dallago, Olenyi, & Rost, 2021; Morton et al., 2020; Villegas-Morcillo et al., 2020).

Although embeddings derived from sequences contain substantially more information than raw sequences, one challenge for this new representation is to simplify its availability. This is one crucial objective of the *bio_embeddings* software resource, which collects tools to create and use protein embeddings. Basic Protocol 1 serves as a high-level overview of functionalities of the *bio_embeddings* pipeline. Basic Protocol 2 adds in-depth context for embeddings and details steps on how to extract embeddings from sequences. Through Basic Protocol 3 (and variations thereof in Alternate Protocols 1 and 2), embeddings are leveraged to plot sequence sets in combination with aspects of protein function, namely subcellular location and membrane-boundness. Finally, in Basic Protocol 4, the rich protein representations from a protein LM are used as input features to train a machine learning device to predict protein subcellular localization.

GENERIC USE OF THE *bio_embeddings* PIPELINE TO PLOT PROTEIN SEQUENCES AND ANNOTATIONS

BASIC PROTOCOL 1

This protocol serves as non-technical overview of what is available out-of-the-box through the *bio_embeddings* pipeline. The premise is simple: you will use software to plot protein sequences and color them by a property. For this purpose, we prepared three files for download: one containing about 100 protein sequences in FASTA format, a CSV file containing DisProt (Hatos et al., 2020) classifications for these sequences (whether their 3D structure presents mostly disorder or little disorder), and a configuration file that specifies parameters for the computation. Apart from downloading these files and the steps necessary to install the *bio_embeddings* software, executing the computation is a single step. The following basic protocols present greater detail about the technical aspects surrounding inputs, outputs and parameters of the pipeline.

The output obtained by us when executing this protocol is available for comparison at *http://data.bioembeddings.com/disprot/disprot_sampled*; the plot file resulting

from executing the steps is available at *http://data.bioembeddings.com/disprot/disprot_sampled/plotly_visualization/plot_file.html*.

NOTE: This visualization is produced for a small sample of DisProt sequences; as such it is by no means representative of the power of the embeddings in distinguishing DisProt classes.

Materials

Hardware

A modern computer (newer than 2012), with about 8 GB of available RAM, 2 GB of available disk space, and an Internet connection.

Software

Windows users may need to install Windows Subsystem for Linux (*https://docs.microsoft.com/en-us/windows/wsl*). All users should have Python 3.7 or 3.8 installed (*https://www.python.org/downloads*).

Data

You will need a FASTA sequence for some proteins in DisProt, which can be downloaded from *http://data.bioembeddings.com/disprot/sequences.fasta*; you will need annotations of disorder content, which can be downloaded from *http://data.bioembeddings.com/disprot/disprot_annotations.csv*; finally, you need a configuration file for the bio_embeddings pipeline, which can be downloaded from *http://data.bioembeddings.com/disprot/config.yml*.

1. Ensure that all software and hardware requirements are met (see Materials, above). Install Python 3.7 or 3.8 on your system using *https://www.python.org/downloads*.

   If you already have a Python installation with a different version (e.g., 2.7) that you must keep, consider installing Python 3.8 through Anaconda ("Anaconda Software Distribution," 2020): https://docs.anaconda.com/anaconda/install.

2. Download required files.
   Through your browser, navigate to *http://data.bioembeddings.com/disprot* and download the files: `sequences.fasta`, `config.yml`, and `disprot_annotations.csv`.

   Note that you might need to right click and select "Save Link As" to download the files.

   If you prefer to use the terminal, run the following three commands:

   ```
   wget http://data.bioembeddings.com/disprot/sequences.fasta
   wget http://data.bioembeddings.com/disprot/config.yml
   wget http://data.bioembeddings.com/disprot/disprot_annotations.csv
   ```

3. Create a project directory and move files into it.
   Create a new directory called `disprot` on your computer and move the files downloaded in step 2 into this directory.

   We suggest creating the directory in an easy-to-find location, for example the `Downloads` folder.

4. Open a new terminal window.
   To open a terminal on MaxOS or Linux, search for the application "Terminal" and open it. On Windows, after having installed the Windows Subsystem for Linux (*https://docs.microsoft.com/en-us/windows/wsl*), search for and open the application called "bash" through the start menu.

5. Install *bio_embeddings*.

To install the pipeline and all of its dependencies, open a terminal window and type in the command:

```
pip install ---user "bio-embeddings[all]"
```

This command may take up to 10 min to execute, depending on the speed of the connection. If you experience warnings regarding incompatible packages (e.g., "`bio-embeddings requires Y>X, but you have Y Z which is incompatible`"), please try using a new conda environment (see Troubleshooting).

6. Navigate to the project directory from the terminal window.
   If you called your project directory `disprot` inside the `Downloads` folder, the command to navigate to the directory through the MacOS and Linux Terminal is:

```
cd ~/Downloads/disprot
```

7. Run the *bio_embeddings* pipeline.
   To start running the *bio_embeddings* pipeline, type the following in your terminal window:

```
bio_embeddings config.yml
```

   Then, press Enter.

   This will start a job using parameters defined in the text configuration file (`config.yml`; detail about the parameters in the next protocols). Opening the file with a text editor will display the following content:

```
global:
   sequences_file: sequences.fasta
   prefix: disprot_sampled
protbert_embeddings:
   type: embed
   protocol: prottrans_bert_bfd
   reduce: true
   discard_per_amino_acid_embeddings: true
umap_projections:
   type: project
   protocol: umap
   depends_on: protbert_embeddings
   n_components: 2
plotly_visualization:
   type: visualize
   protocol: plotly
   annotation_file: disprot_annotations.csv
   display_unknown: false
   depends_on: umap_projections
```

   There are four major text blocks, each defining a job stage. The parameters in the first block (starting with `general`) define where protein sequences live and where to store results. The second block (`protbert_embeddings`) defines parameters to generate computational representations using a language model (more in the following). The third (`uma_projections`) contains options to transform the representations, while the forth (`plotly_visualizations`) defines options to plot the proteins.

   You should see output that resembles:

```
2020-11-09 20:37:13,753 INFO Created the prefix directory disprot_sampled
2020-11-09 20:37:13,756 INFO Created the file
disprot_sampled/input_parameters_file.yml
2020-11-09 20:37:13,970 INFO Created the file disprot_sampled/sequences_file.fasta
2020-11-09 20:37:14,118 INFO Created the file disprot_sampled/mapping_file.csv
…
```

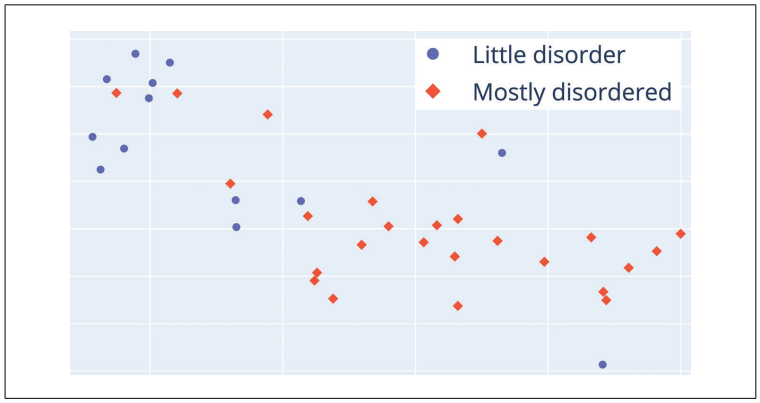


Figure 1 2D visualization of protein sequences with disorder annotation. The points are projections of embeddings of a subset of protein sequences contained in DisProt (Hatos et al., 2020). Proteins annotated with high disorder content (red) tend to cluster to the bottom-right, while proteins annotated with little disorder content (blue) tend to cluster to the top-left. The figure is available interactively at *http://data.bioembeddings.com/figures/figure_1.html*.

Please note that sometimes warnings may appear as dependencies used by the bio_embeddings pipeline get updated and introduce slight changes in how bio_embeddings is expected to interface with them. Warnings are usually harmless and get addressed by the bio_embeddings team within a few weeks. The command will take up to 15 min to execute and will download a 1.5-GB file in your home directory.

8. Open the plot file.

After the execution of the bio_embedding pipeline has finished, your system should automatically have opened up a browser window displaying a 2D graph of the proteins contained in the FASTA file colored by their disorder content according to DisProt (Hatos et al., 2020; Fig. 1). If not, you can navigate to the `disprot` *directory, which will contain a new directory (*`disprot_sampled`*), with yet another directory (*`plotly_visualization`*), which contains the plot file as* `plot_file.html` *. You can open this file in any modern browser.*

BASIC PROTOCOL 2

GENERATE EMBEDDINGS FROM PROTEIN SEQUENCES USING THE *bio_embeddings* PIPELINE

Through this protocol, you may generate machine-readable representations (embeddings) from a set of protein sequences using the "embed" stage of the *bio_embeddings* pipeline. The sequence file utilized for the example was written by the prediction program DeepLoc (Almagro Armenteros, Sønderby, Sønderby, Nielsen, & Winther, 2017), but you can also provide your own FASTA file. Embeddings constitute an abstract encoding of the information contained in protein sequences, and are the building block of the pipeline and its analytical tools. In this protocol, we use BERT (Devlin, Chang, Lee, & Toutanova, 2019) trained on BFD (Steinegger & Söding, 2018; Steinegger et al., 2019) to extract embeddings from protein sequences. This model is part of the ProtTrans protein LMs (Elnaggar et al., 2020), referred to as ProtBERT in text or `prot_trans_bert_bfd` in the following code. You can find out how to choose a protein LM based on your requirements on our website (*http://bioembeddings.com*). The salient output of the *embed* stage are the embedding files. These come in two flavors: per-residue (`embeddings_file.h5`) and per-protein (`reduced_embeddings.h5`). While the per-residue embeddings are taken directly out of the LMs, per-protein embeddings are generated post-processing the information extracted by the LM through global average

pooling (Shen et al., 2018) on all combined per-residue embeddings of a sequence. Per-residue embeddings are useful to analyze properties of residues in a protein (e.g., which residues bind ligands), while per-protein representations capture annotations describing entire proteins (e.g., native localization).

Materials

Hardware

Computer (newer than 2012), >8 GB of available RAM, ∼2 GB of available disk space
Optional: Graphical Processing Unit (GPU) with >4 GB of vRAM and supporting CUDA® 11.0
This will speed up the embedding process manyfold
Internet connection

Software (MacOS and Linux)

Python 3.7 or 3.8 (*https://www.python.org/downloads*)
Windows users: Windows Subsystem for Linux (*https://docs.microsoft.com/en-us/windows/wsl*)
Optional: CUDA® (*https://developer.nvidia.com/cuda-downloads*; at time of writing: version 11.1)

Data

DeepLoc (Almagro Armenteros et al., 2017): *http://data.bioembeddings.com/deeploc/deeploc_data.fasta*
DeepLoc (reduced sample) FASTA-formatted sequences: *http://data.bioembeddings.com/deeploc/sampled_deeploc_data.fasta*
NOTE: as input, we begin with two files containing protein sequences in a simplified FASTA format (first line begins with ">" followed by protein name, all subsequent lines contain the sequence in single-letter amino acid code).

1. Install *bio_embeddings* from pip.
   To install the pipeline and all of its dependencies, open a terminal window and type in the command:

   ```
   pip install --user "bio-embeddings[all]"
   ```

2. Create a project directory.
   We suggest you create a new project directory on your disk. You can generate it through the terminal:

   ```
   mkdir deeploc
   ```

   Then, open the directory through the terminal:

   ```
   cd deeploc
   ```

3. Download the DeepLoc FASTA file inside the project directory.
   From the terminal (within the project directory):

   ```
   wget http://data.bioembeddings.com/deeploc/deeploc_data.fasta
   ```

   Alternatively, download the file using your browser, and move it to the project directory.

CAUTION: *If you are using a system not equipped with a GPU, we suggest picking a smaller FASTA set for the next steps. This will facilitate executing subsequent steps. A smaller FASTA file is available at: http://data.bioembeddings.com/deeploc/sampled_deeploc_data.fasta. If you pick this file, make sure to note the name change for the following steps.*

4. Create a configuration file.
   A configuration file defines what the pipeline should do (files and parameters it should use and stages it should run). Many examples of configuration files are provided at *http://examples.bioembeddings.com*, including the one you will create here (called `deeploc`). To create the configuration file from the terminal:

```
nano config.yml
```

   Then, type in the following and save the file (to save: press Ctrl+x, then "y", then the Return key):

```
global:
   sequences_file: deeploc_data.fasta
   prefix: deeploc_embeddings
   simple_remapping: True

prottrans_bert_embeddings:
   type: embed
   protocol: prottrans_bert_bfd
   reduce: True
```

   The `global` section defines a global parameter; mandatory are the input sequence file (called `deeploc_data.fasta` in the config) and the prefix where outputs will be stored (in this case, a new directory `deeploc_embeddings`, which will be created inside the deeploc project directory).

   The sections following `global` define stages of the pipeline and can have arbitrary names. In this case, you have one stage called `prottrans_bert_embeddings`, which will execute an "embed" stage (`type: embed`), using the BERT language model trained on BFD (Elnaggar et al., 2020) (`protocol: prottrans_bert_bfd`). The "embed" stage produces per-residue embeddings by default. To get per-protein embeddings you must specify the `reduce` parameter (`reduce: True`).

5. Run the *bio_embeddings* pipeline.
   All that is left to do is to supply the configuration file to bio_embeddings and let the pipeline execute the job. To do so, type on the terminal:

```
bio_embeddings config.yml
```

   You should see output that resembles:

```
2020-11-09 20:37:13,753 INFO Created the prefix directory deeploc_embeddings
2020-11-09 20:37:13,756 INFO Created the file deeploc_embeddings/input_parameters_file.yml
2020-11-09 20:37:13,970 INFO Created the file deeploc_embeddings/sequences_file.fasta
2020-11-09 20:37:14,118 INFO Created the file deeploc_embeddings/mapping_file.csv
…
```

6. Locate the embedding files.

   After the job has finished, you should have a new directory called `deeploc_embeddings` (prefix) in your `deeploc` project directory. This directory will contain several files, and another directory, `prottrans_bert_embeddings` (`config.yml` after section `global`), with the outputs of the "embed" stage. The most salient files are `embeddings_file.h5` and `reduced_embeddings_file.h5` (only produced if "`reduce: True`") inside the `prottrans_bert_embeddings` directory. These files are what you will use for your analyses and to train prediction tools (following protocols).

OVERLAY SEQUENCE ANNOTATIONS ON A PROTEIN SPACE VISUALIZATION

The previous protocol generated embeddings from protein sequences in your dataset (here DeepLoc dataset). In Basic Protocol 3 you use functions from the *bio_embeddings* package to visualize "protein spaces" spanned by the embeddings extracted. These visualizations reveal whether or not the LM chosen for the "embed" stage (Basic Protocol 2) can roughly separate your data based on a desired property/phenotype. The property/phenotype in our example is subcellular location in 10 states. Alternate Protocol 2 uses the same data and similar steps to visualize protein solubility. While visualizations are useful, the discriminative power of embeddings can be boosted many times by training machine learning models on the embeddings to predict the desired property (Basic Protocol 4).

Between *embedding generation* and *protein space visualization*, another step has to be inserted. In the pipeline, we refer to this step as a "project" stage. Its purpose is to reduce the dimensionality of the embeddings (e.g., 1024 for ProtBERT) such that it can be visualized in either 2D or 3D. Here, we project embeddings onto 2D; Alternate Protocol 1 uses the same data and slight variations in parameters to 3D plots instead.

The final notebook constructed here is available at *http://notebooks.bioembeddings.com* as `deeploc_visualizations.ipynb` to be downloaded and executed locally, or executed directly online. The file also includes steps presented in Alternate Protocols 1 and 2.

The Support Protocol 1 explains how to integrate the final visualization options in a configuration file as instruction for the pipeline to manage the entire process—from sequences to visualizations. This is useful to enable colleagues to reproduce all your results from a few files.

Materials

Software

Jupyter Notebook (Kluyver et al., 2016)

Notebooks can be run locally, provided that the necessary dependencies are installed (python 3.7 and the Jupyter suite). Installation steps are described here: https://jupyter.org/install.

Notebooks can be run on Google Colaboratory (Bisong, 2019), without having to install software locally, given an internet connection and a Google account.

Data

DeepLoc embeddings input files, which you either calculated through Basic Protocol 2 or you can be download from *http://data.bioembeddings.com/deeploc/reduced_embeddings_file.h5*

Annotations of properties/phenotypes of the proteins; for DeepLoc, subcellular location annotations can be downloaded from *http://data.bioembeddings.com/deeploc/annotations.csv*

1. Create new Jupyter Notebook on Google Colaboratory (a) or locally (b).

   a. We suggest running the following through Google Colaboratory. To open a new Google Colaboratory, navigate to: *https://colab.research.google.com/#create=true*.

b. If you prefer to execute the steps on your local computer, through the terminal, navigate to the `deeploc` folder created previously, or to a new folder. Then, start a Jupyter notebook through the terminal:

```
jupyter-notebook
```

This should open a browser window. From the top-right drop-down menu called "new", select "Python 3".

2. Install bio_embeddings

   a. On Google Colaboratory paste in the following code in the first code block:

```
!pip3 install -U pip
!pip3 install -U "bio-embeddings[all]"
```

   Then, press the play button on the left of the code cell. Given some version differences in Google Colaboratory, warnings may arise. These, however, can be ignored.

   b. If you already executed Protocol 1, you are set. Otherwise, open a new terminal window and type:

```
pip install --user "bio-embeddings[all]"
```

3. Download files.

   a. On Google Colaboratory, create a new code block (by pressing the "+ code" button). Then, paste in the following code:

```
!wget http://data.bioembeddings.com/deeploc/reduced_embeddings_file.h5
!wget http://data.bioembeddings.com/deeploc/annotations.csv
```

   b. On your local computer, simply download the files listed in the Materials list for this protocol and move them into the folder in which the notebook was started (see step 1).

4. Import dependencies.
   From here on, the execution steps are identical on Google Colaboratory and your local Jupyter notebook. You will now import the functions that allow you to open embedding files, reduce the dimensionality, and visualize scatter plots. To do so, in a new code block, type and execute the following:

```
import h5py
import numpy as np
from pandas import read_csv, DataFrame
from bio_embeddings.utilities import QueryEmbeddingsFile
from bio_embeddings.project import umap_reduce
from bio_embeddings.visualize import render_scatter_plotly
```

5. Read annotations file.
   Assume that the original FASTA file, for which you generated embeddings, was the following:

```
>Q9H400-2
SEQVENCE
>P12962
SEQVVNCE
>P12686
MNQVENCE
```

You can define a set of annotations for the sequences in this set as a CSV file, containing minimally two columns called "identifier" and "label" such as:

```
identifier,label
Q9H400-2,Cell membrane
P12962,Cytoplasm
P12686,Mitochondrion
```

The identifiers have to match to the identifiers in the FASTA header of the protein sequences for which embeddings have been computed. They can, however, only contain a subset of identifiers with respect to the embeddings.
You can now load the `annotations.csv` file which we have created based on the DeepLoc data. These annotations contain experimentally validated subcellular location in 10 classes. To load them into the notebook, execute the following in a new code block:

```
annotations = read_csv('annotations.csv')
```

6. Read the embeddings file.
   In a new code block, type and execute the following:

```
identifiers = annotations.identifier.values
embeddings = list()

with h5py.File('reduced_embeddings_file.h5', 'r') as embeddings_file:
   embedding_querier = QueryEmbeddingsFile(embeddings_file)

   for identifier in identifiers:
      embeddings.append(embedding_querier.query_original_id(identifier))
```

   This will store the embeddings in the "embeddings" list in the same order as the identifiers. To access the embeddings, you can use a helper class called "QueryEmbeddingsFile". This class allows you to retrieve embeddings either using the identifier extracted from the FASTA header (as done here, via the `query_original_id` function), or by using the pipeline's internal identifier for protein sequences. You can find more information about these functions at https://docs.bioembeddings.com.

7. Project embeddings to 2D using UMAP (McInnes, Healy, & Melville, 2018).
   In a new code block, type and execute the following:

```
options = {
   'min_dist': .1,
   'spread': 8,
   'n_neighbors': 160,
   'metric': 'euclidean',
   'n_components': 2,
   'random_state': 10
}
projected_embeddings = umap_reduce(embeddings, **options)
```

   This code block will take some minutes to execute (4 min on Google Colaboratory), as projecting the embeddings is a compute-intensive operation. Projecting embeddings onto fewer dimensions is necessary because data in dimensions d>3 is very tricky to plot (and even d = 3, i.e., 3D plots of scientific data, are often difficult to grasp quickly). RAW embeddings have much higher dimensions, e.g., d = 1024 dimensions for ProtBERT (Elnaggar et al., 2020). In "options", you can define UMAP parameters. These parameters can be tuned to generate different visualizations, e.g., you could change the "metric" to "manhattan". To graphically see the effect of changing options, you may execute the steps from here onward again. The "`projected_embeddings`" variable contains a Numpy (Harris

Table 1 Example of Merged Annotations and Projected Embeddings

Identifier	Label	Component_0	Component_1
Q9H400	Cell.membrane	2.474637	–8.919042
Q5I0E9	Cell.membrane	32.507015	10.355012
P63033	Cell.membrane	18.500378	–0.299981
Q9NR71	Cell.membrane	2.420154	18.161064
Q86XT9	Cell.membrane	–4.937888	–1.767011

et al., 2020) matrix of size N×2, where N is the number of proteins for which there are embeddings in the embedding file, while 2 is dictated by the "`n_components`" in "`options`" (number of output dimensions of the projection).

8. Merge projected embeddings and annotations.
   In a new code block, type and execute the following:

```
projected_embeddings_dataframe = DataFrame(
   projected_embeddings,
   columns=["component_0", "component_1"],
   index=identifiers
)
merged_annotations_and_projected_embeddings = annotations.join(
   projected_embeddings_dataframe, on="identifier", how="left"
)
```

   Here, you create a DataFrame (similar to a table) from the projected embeddings. Rows are indexed by the "`identifiers`", while the two columns contain the two components of te projected embeddings. In other words: you are constructing a table of coordinates for your protein sequences. Lastly, you merge these coordinates with the annotations. You can inspect the first five rows of the dataframe by typing the following into a new code block and executing it:

```
merged_annotations_and_projected_embeddings[:5]
```

   This should resemble the content reported in Table 1.

9. Plot the protein space spanned by the projected embeddings
   In a new code block, type and execute the following:

```
figure = render_scatter_plotly(merged_annotations_and_projected_embeddings)
figure.show()
```

   This will display an interactive plot (of which a static screenshot is provided in Fig. 2). Interactive plots make it possible to disentangle complex annotations/datasets, e.g., by toggling the display of some annotations (click on the legend). Even more useful: zoom in and out plots, especially when visualizing 3D plots.

TRAIN A MACHINE LEARNING CLASSIFIER ON PROTEIN EMBEDDINGS

Basic Protocol 2 generated embeddings for proteins in DeepLoc (Almagro Armenteros et al., 2017). Basic Protocol 3 visualized the projected embeddings in a 2D plot and annotated the proteins in this 2D plot by colors signifying subcellular location. In the following

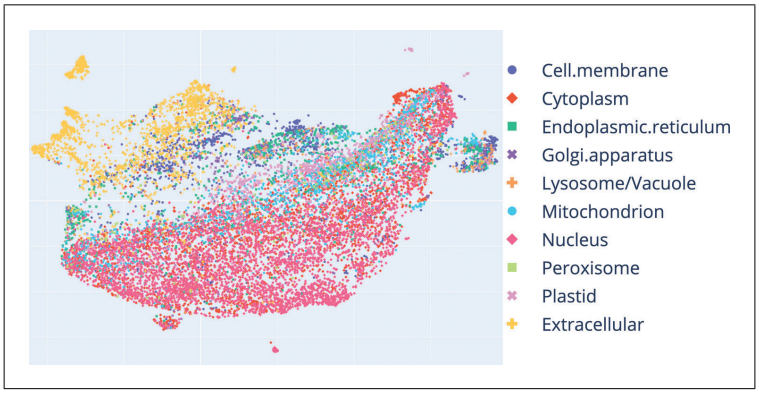


Figure 2 2D protein space drawn by projected DeepLoc embeddings. Points are projections of embeddings of protein sequences in the DeepLoc set (Almagro Armenteros et al., 2017). Coloring is provided according to their subcellular location. Of note: "Extracellular" proteins seem to be particularly keen on forming a cluster, while proteins in other localizations barely separate into groups inside a bigger cluster. The figure is available interactively at: *http://data.bioembeddings.com/figures/figure_2.html*.

steps, you will use the embeddings generated through the pipeline and the location annotations from DeepLoc to machine-learn the prediction of location from protein sequence embeddings. Once trained, you can apply this prediction method to annotate/predict location for any protein sequence. The simplest recipe to build a generic machine learning model is as follows:

1. Divide data into train and test sets (these should be sequence-non-redundant with respect to each other, i.e., no protein sequence in one should be more sequence-similar than some threshold to any protein in the other; what this threshold is depends on your task)
2. Split a subset from the train set to construct a validation set (non-redundant to split-off)
3. Evaluate some machine learning hyper-parameters using the validation set (e.g., which type of machine learning model—such as ANN, CNN, or SVM, what particular choice of parameters—such as number of hidden units/layers for ANN/CNN). Construct a leaderboard (i.e., a table keeping track of the relative performance of all the models/hyper-parameters).
4. Select the best model from the leaderboard, and evaluate on the test set (by NO MEANS apply all models to the test set and pick the best; instead, it is essential to choose the best using the validation set and to stick to that choice to avoid over-fitting).
5. Report performance for a diversity of relevant evaluation metrics for the final model using the test set (include estimates for standard errors)

The following steps explore this recipe using sci-kit learn (Pedregosa et al., 2011). You will produce a classifier which roughly separates the ten location classes from DeepLoc (Almagro Armenteros et al., 2017). The objective of this protocol is not to produce the best prediction method for subcellular location classification, which would require more parameter testing and tuning! Instead, the objective is to showcase the ease of going from data to prediction method when using embeddings. The final notebook constructed here is available at *http://notebooks.bioembeddings.com* as downloadable file called `deeploc_machine_learning.ipynb`.

Materials

See Basic Protocol 3

1. Complete steps 1-5 of Basic Protocol 3.
2. Import additional dependencies.
   Via a new code block, you will import a set of dependencies from the popular machine learning library scikit-learn (Pedregosa et al., 2011) in order to train and evaluate the machine learning model:

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score
```

3. Split annotations into train and test sets

   The first task for any supervised machine learning is the split of the data into training and testing sets. The testing set (also referred to as "hold out set") is used exclusively to evaluate the performance of the final machine learning model. The training set serves the optimization of the model and hyper-parameters.

   In computational biology/bioinformatics, informed decisions on how to split data are pivotal, for example, by ascertaining that no protein in the training set has more than 20% pairwise sequence identity (PIDE) to any protein in the test set (Reeb, Goldberg, Ofran, & Rost, 2020). While packages such as scikit-learn (Pedregosa et al., 2011) include functions to easily split data into train and test sets, they completely fail to account for domain knowledge such as the concept of homology or evolutionary connections relevant to reduce redundancy between bio-sequences. Therefore, users of such packages have to address these issues manually when starting a new project, or they will join the many who produce overconfident methods.

   DeepLoc annotations come with a column "set" which is either "train" or "test". The split into these two categories has been made such that any pair of sequences in train and test share at most 30% PIDE. To split the data, execute the following block of code:

```
train_set = annotations[annotations.set == "train"]
test_set = annotations[annotations.set == "test"]
```

4. Load embeddings into train and test sets.
   Once you have split the annotations into train and test sets, you need to create input and output for the machine learning model. The input will be the sequence embeddings (in the following, "`training_embeddings`"), while the output will be the subcellular location associated to those proteins (in the following, "`training_labels`"). In a new code block, type the following:

```
training_embeddings = list()
training_identifiers = train_set.identifier.values
training_labels = train_set.label.values

testing_embeddings = list()
testing_identifiers = test_set.identifier.values
testing_labels = test_set.label.values

with h5py.File('reduced_embeddings_file.h5', 'r') as embeddings_file:
    embedding_querier = QueryEmbeddingsFile(embeddings_file)

    for identifier in training_identifiers:
        training_embeddings.append(embedding_querier.query_original_id(identifier))

    for identifier in testing_identifiers:
        testing_embeddings.append(embedding_querier.query_original_id(identifier))
```

5. Define basic machine learning architecture and parameters to optimize
   In a new code block, type and execute the following:

```
multilayerperceptron = MLPClassifier(
   solver='lbfgs',
   random_state=10,
   max_iter=1000
)
parameters = {
    'hidden_layer_sizes': [(30,), (20,15)]
}
```

This will create a basic neural network architecture ("`multilayerperceptron`") and a set of parameters that you want to test during parameter optimization. The basic architecture uses the "Limited-memory Broyden–Fletcher–Goldfarb–Shanno Algorithm" solver (Saputro & Widyaningsih, 2017) and a maximum of 1000 training iterations (`max_iter`). Using the "lbfgs" solver, maximum training iterations correspond to how many embeddings the algorithm will maximally see before training is stopped. Training may automatically be stopped before the maximum number of iterations if the model converges (in other words: if its validation error stays within a certain threshold). In the DeepLoc set, there are more than ten thousand samples, so `max_iter` could be set to a higher value, but for the purpose of this protocol, to have reasonable execution time, we propose limiting the number of iterations to 1000.

The parameter that you will optimize is the number of hidden layers and the amount of neurons in each layer. In one case, you will try a network with one hidden layer containing 30 neurons, while in the second case you will test a network with two hidden layers with 20 and 15 neurons, respectively.

6. Train classifiers and pick the best performing model.

Usually, this step is performed in various sub-steps, for example: first you define the number of training splits (e.g., Nsplit=3), which would give you data for training (optimization of free parameters) and for cross-training/validation (optimization of hyper-parameters and model choice). Then, you train Nsplit-1 (i.e., 2 for Nsplit=3) network variants describing each split, evaluate on the respective validation data, and finally select the network performing best (on the cross-training/validation split). Luckily, all of these steps can be summarized into three lines of code using sci-kit learn. For this example, we have ignored homology/redundancy when splitting the data set for brevity, but in real-life applications, accounting for homology/redundancy when splitting is essential to obtain valid models!

   In a new block of code, write and execute the following:

```
classifiers = GridSearchCV(
   multilayerperceptron,
   parameters, cv=3,
   scoring="accuracy"
)
classifiers.fit(training_embeddings, training_labels)
classifier = classifiers.best_estimator_
```

Note this code takes about 15 min to execute on Google Colab. No output is produced during this time. Visual clues from the notebook assist you in understanding when the computation is over. Another important note on scope: while you will obtain a classifier that is roughly able to classify sequences in ten subcellular location compartments, your method will not beat the state-of-the-art for this problem due to extensive development in the field! The goal of this protocol is to give you the tools to build a classifier, as well as to require little time to execute. If you want to obtain the best classifier, you will need to test and tune more parameters, and especially consider more training iterations (as defined by `max_iter` in the previous step).

7. Predict subcellular location for test set and calculate performance,
   Lastly, to evaluate the performance of you final model, you predict the location for all proteins in the test set and calculate accuracy as follows:

```
predicted_testing_labels = classifier.predict(testing_embeddings)
accuracy = accuracy_score(
   testing_labels,
   predicted_testing_labels
)

print(f"Our model has an accuracy of {accuracy:.2}")
```

   The reported accuracy should be 0.72.

8. *Optional:* Embed a novel sequence and predict its subcellular location.
   In this optional step, you generate the sequence embedding for an arbitrary sequence and use the classifier developed in the previous steps to predict its subcellular location. To do so, type and execute the following:

```
from bio_embeddings.embed import ProtTransBertBFDEmbedder

embedder = ProtTransBertBFDEmbedder()

sequence = "DDCGKLFSGCDTNADCCEGYVCRLWCKLDW"
per_residue_embedding = embedder.embed(sequence)
per_protein_embedding = embedder.reduce_per_protein(per_residue_embedding)
sequence_subcellular_prediction = classifier.predict([per_protein_embedding])[0]

print("The arbitrary sequence is predicted to be located in: "
   f"{sequence_subcellular_prediction}")
```

   Above, you import the "`ProtTransBertBFDEmbedder`" and initialize it. You then define an amino acid sequence using the standard IUPAC alphabet. The sequence is then embedded per-residue (`per_residue_embedding`), and the per-residue embedding is transformed to a per-protein embedding via a helper function (`per_protein_embedding`). Finally, the per-protein embedding is used to predict subcellular location through the classifier you developed, and the prediction (Extracellular) is printed to screen.

   You may see a warning about "padding" appear in the output; you can ignore this as it will not affect execution.

   For scikit-learn the function "`predict`" expects a list of protein embeddings. This (usually helpful) feature implies that additional steps are required to predict for a single sequence, namely that first you have to put the embedding into a list. You can then grab the prediction of the first (and only) item in the list, which will be the prediction of the arbitrary sequence.

ALTERNATE PROTOCOL 1

GENERATE 3D INSTEAD OF 2D VISUALIZATIONS

The following steps introduce minimal code changes with respect to the steps and code outlined in Basic Protocol 3 to visualize in 3D instead of 2D. We assume that the code from Basic Protocol 3 has been written in a Jupyter/Colab Notebook and highlight code changes in orange. Visit the docs at *https://docs.bioembeddings.com* to find out more about the functions of the *bio_embeddings* package.

The code from Basic Protocol 3 is available at *http://notebooks.bioembeddings.com* as downloadable file called `deeploc_visualizations.ipynb`. It includes the steps presented here in an alternate form.

Materials

See Basic Protocol 3

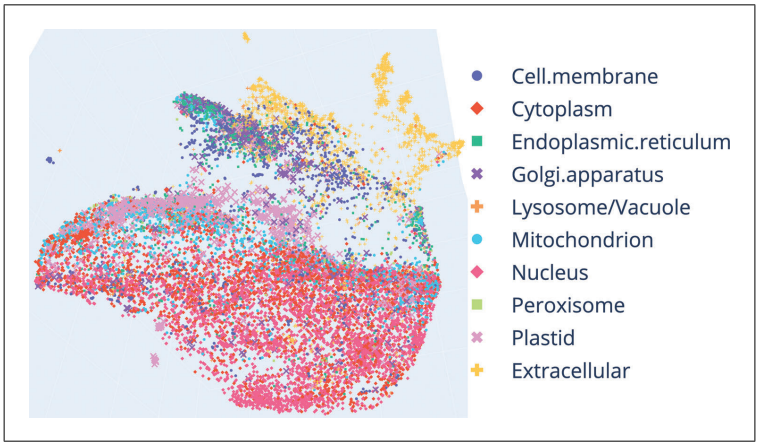


Figure 3 3D protein space drawn by projected DeepLoc embeddings. Points are projections of embeddings of protein sequences in the DeepLoc set (Almagro Armenteros et al., 2017). Coloring is provided according to their subcellular localizations. The 3D figure is best explored interactively: *http://data.bioembeddings.com/figures/figure_3.html*.

1. Project embeddings onto 3D instead of onto 2D.

   The first change to the previous steps requires only augmenting the number of components UMAP will project embeddings to.

   Take the code block written in Basic Protocol 3, step 7, and locate and change the line:

   ```
   'n_components': 2
   ```

   to:

   ```
   'n_components': 3
   ```

   Then, re-run the code cell.

2. Import 3D scatter plot renderer instead of 2D.
   Change the import of the visualization function from Basic Protocol 3, step 4, from:

   ```
   from bio_embeddings.visualize import render_scatter_plotly
   ```

   to:

   ```
   from bio_embeddings.visualize import render_3D_scatter_plotly
   ```

   and execute the code block.

3. Add a third component to the projected embeddings DataFrame.
   Change the number of components in the projected DataFrame defined in B.asic Protocol 3, step 8 from:

   ```
   columns=["component_0", "component_1"],
   ```

to:

```
columns=["component_0", "component_1", "component_2"],
```

and execute the code block.

4. Swap the plotting function with the 3D variant:
   Lastly, swap out the plotting function name in the code block created in Basic Protocol 3, step 9, from:

```
figure = render_scatter_plotly(

   merged_annotations_and_projected_embeddings
)
```

to:

```
figure = render_3D_scatter_plotly(

   merged_annotations_and_projected_embeddings
)
```

and execute the code block.

At this point, a 3D interactive plot (Fig. 3) will be displayed on your notebook.

ALTERNATE PROTOCOL 2

VISUALIZE CLASSIFICATION INTO MEMBRANE/SOLUBLE INSTEAD OF PROTEIN SUBCELLULAR LOCATION

The following steps introduce minimal code changes with respect to the steps and code outlined in Basic Protocol 3 in order to visualize the classification into membrane/soluble proteins as annotated in DeepLoc (Almagro Armenteros et al., 2017) instead of location. We assume that the code from Basic Protocol 3 has been written up and highlights code changes in orange.

The code from Basic Protocol 3 is available at *http://notebooks.bioembeddings.com* as downloadable file called `deeploc_visualizations.ipynb`. It includes the steps presented here in an alternate form.

Materials

Software and Hardware

See Basic Protocol 3

Data

DeepLoc solubility annotations: *http://data.bioembeddings.com/deeploc/solubility_annotations.csv*

1. Download additional file `solubility_annotations.csv`.

   a. On Google Colaboratory create a new code block (by pressing the "+ code" button). Then, paste in the following code:

```
!wget http://data.bioembeddings.com/deeploc/solubility_annotations.csv
```

b. On your local computer, simply download the file listed in the Materials list for this protocol and move into the folder in which the notebook was started (see Basic Protocol 3, step 1).

2. Change the annotations file.
In the code block created in Basic Protocol 3, step 5, change the input file from:

```
annotations = read_csv('annotations.csv')
```

to:

```
annotations = read_csv('solubility_annotations.csv')
```

3. Re-run the subsequent code blocks.

Re-run every code block following the code block just changed. This will display a graph, this time colored according to protein solubility, i.e., whether a protein is annotated as membrane-bound, soluble or lacks an annotation).

PUT EMBEDDING GENERATION AND SEQUENCE SPACE VISUALIZATIONS TOGETHER IN ONE PIPELINE

SUPPORT PROTOCOL

Basic Protocol 3 presents an explorative approach towards producing protein-space visualizations. In this Support Protocol, you will use the parameters chosen in Basic Protocol 3 to define a pipeline configuration file. These files allow reproducible workflows. You will do so by extending the *bio_embeddings* configuration presented in Basic Protocol 2, step 4, to also generate protein space visualizations. Noteworthy differences with previous files will be highlighted in orange.

Materials

Software and Hardware

See Basic Protocol 2

Data

DeepLoc FASTA file: *http://data.bioembeddings.com/deeploc/deeploc_data.fasta*
DeepLoc subcellular location annotations: *http://data.bioembeddings.com/deeploc/annotations.csv*

1. Execute steps 1 through 3 of Basic Protocol 2.

2. Download the annotations file into the project directory.
From the terminal (within the project folder):

```
wget http://data.bioembeddings.com/deeploc/annotations.csv
```

Alternatively, download the file using your browser (link in the Materials of this protocol), and move it to the project directory.

3. Define a configuration file to embed, project and visualize protein sequences.
Similarly to Basic Protocol 2, step 4, we define a text file (`config.yml`) that contains the following text:

```
global:
   sequences_file: deeploc_data.fasta
   prefix: deeploc_embeddings
   simple_remapping: True

prottrans_bert_embeddings:
   type: embed
   protocol: prottrans_bert_bfd
   reduce: True
   discard_per_amino_acid_embeddings: True

umap_projections:
   type: project
   protocol: umap
   depends_on: prottrans_bert_embeddings
   min_dist: 0.1
   spread: 8
   n_neighbors: 160
   metric: euclidean
   n_components: 2
   random_state: 10

plotly_visualization:
   type: visualize
   protocol: plotly
   depends_on: umap_projections
   annotation_file: annotations.csv
   display_unknown: False
```

The first part of this config (“`global`” and “`prottrans_bert_embeddings`”) are almost identical to the config presented in Basic Protocol 2. The addition of the “`discard_per_amino_acid_embeddings`” parameter tells the pipeline that we are only interested in the per-protein embeddings (`reduced_embeddings_file.h5`), and that the per-residue embeddings (`embedding_file.h5`) should not be stored on disk. This will save significant storage space.

A stage (`umap_projections`) of type “`project`” that uses the protocol umap was added. The “`depends_on`” directive tells the pipeline that the embeddings generated by “`prottrans_bert_embeddings`” should be used for the project stage. We add the same UMAP parameters as in Basic Protocol 3, step 7. This stage will output a DataFrame of the projected embeddings (`projected_embeddings.csv`).

Finally, we use this data for a “`visualize`” type stage (by depending on the umap_projections). We annotate the visualization using the annotation file called “`annotations.csv`”. Sequences without annotations (but that might be present in the input FASTA file) will not be plotted (“`display_unknown: False`”). The “`plotly_visualization`” stage will produce a file containing the 2D interactive figure (figure.html).

4. Run the *bio_embeddings* pipeline.
   What remains is to supply the configuration file to bio_embeddings and let the pipeline execute the job. For that type into the terminal:

```
bio_embeddings -o config.yml
```

The “-o” option instructs the pipeline to overwrite a previous pipeline run at the same prefix, which might have remained in the current project directory (`deeploc`) from the previously executed Basic Protocol 2.

5. Locate the interactive figure file.

After the job has finished, you should see a “`deeploc_embeddings`” directory in your project directory. This directory will contain three subdirectories called: `prottrans_bert_embeddings`, `umap_projections`, and `plotly_visualization`. Each directory contains the output of the corresponding stage. The newly created interactive figure will be stored in the

"plotly_visualization" directory as "figure.html". You can use a browser, such as Safari, to open this figure. It should resemble Figure 2.

COMMENTARY

Background Information

Language Models (LMs) such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020), and T5 (Raffel et al., 2020) improve over previous methods for learning to embed text (Bojanowski, Grave, Joulin, & Mikolov, 2017; Mikolov, Chen, Corrado, & Dean, 2013; Pennington, Socher, & Manning, 2014) by cleverly modeling context ("apple" company vs. fruit) and training on increasingly larger natural language corpora. They begin to suggest large models from artificial intelligence (AI) or machine learning (ML) to compete with human experts, at least for some tasks (Manning, 2011). They also help rising questions about current benchmarks (Heinzerling, 2020; McCoy, Pavlick, & Linzen, 2019) and the extent to which LMs truly understand language (Bender & Koller, 2020). Despite potential performance overestimates, LMs succeed to effectively translate natural language besting expert-based models, i.e., they captured the meaning in text automatically (Pires, Schlinger, & Garrette, 2019; Zhu et al., 2020).

Training LMs requires very large amounts of intrinsically structured, sequential data, making these approaches especially promising for ambitious attempts that try to automatically understand the language of life proxied by protein sequences (Heinzinger et al., 2019). In fact, the amount of data available for protein sequences is 500 times larger than the largest NLP data sets such as Google's Billion Word data (Chelba et al., 2014; Steinegger & Söding, 2018; Steinegger et al., 2019). With the increasing degree to which the speed of adding new protein sequences outpaces the improvement in computer hardware, experimental annotations—although also increasing exponentially—cannot keep track with this explosion. Therefore, the sequence-annotation gap, i.e., the gap between the number of proteins with known sequence and those with known annotation, continues to rise.

In analogy to natural languages, protein sequences are formed by tokens (proteins: amino acids, text: words) that have individual and context-dependent meaning through long- and short-range dependencies (proteins: inter-residue bonds, text: sentences). Thus, similarly to natural language, LMs trained on protein sequences (Alley, Khimulya, Biswas, AlQuraishi, & Church, 2019; AlQuraishi, 2019; Armenteros, Johansen, Winther, & Nielsen, 2020; Elnaggar et al., 2020; Heinzinger et al., 2019; Lu, Zhang, Ghassemi, & Moses, 2020; Madani et al., 2020; Min, Park, Kim, Choi, & Yoon, 2020; Rao et al., 2019; Rives et al., 2019) capture important meaning of the protein sequence language, as demonstrated by their ability to predict aspects of protein structure and function. For instance, SeqVec (Heinzinger et al., 2019) trained ELMo (Peters et al., 2018) on UniRef50 (The UniProt Consortium, 2019) and showed that the LM's representations clustered protein sequences by function (Heinzinger et al., 2019). In another analogy to NLP, protein LMs may be fine-tuned on specialized sequence sets (analogy to natural language: legal text vs. wikipedia articles) to encode for different protein properties (Armenteros et al., 2020).

Previously, machine-learning methods in computational biology leveraged data-driven protein representations such as substitution matrices, capturing biophysical features (Henikoff & Henikoff, 1992), family-specific profiles (Stormo et al., 1982), or evolutionary couplings (Morcos et al., 2011) that capture evolutionary features. Now, embeddings provide competitive results for many prediction tasks (Littmann et al., 2021; Rao et al., 2019, 2020). Protein LMs may even be combined with other representations to gain even better performance (Rives et al., 2019; Villegas-Morcillo et al., 2020). Protein sequence embeddings are generated in a fraction of the time it takes to generate MSAs (Heinzinger et al., 2019), and can thus be used on entire proteomes, where MSA-based approaches might be computationally prohibitive or even unavailable (e.g., small protein families).

The *bio_embeddings* pipeline, which is used throughout the manuscript to generate and leverage protein embeddings, is targeted to computational biologists and aims to abstract, via a uniform and standardized interface, the use of protein LMs. Embeddings can be used to train machine learning algorithms using "transfer learning" (Basic Protocol 4; Raina, Battle, Lee, Packer, & Ng, 2007), or for analytical purposes. The pipeline enables visual analysis of sequence sets by drawing protein spaces spawned by

their embeddings (Basic Protocol 3). Users can create representations from a growing diversity of protein LMs, which at the time of writing include: SeqVec (Heinzinger et al., 2019), UniRep (Alley et al., 2019), ESM (Rives et al., 2019), ProtBERT, ProtALBERT, ProtXLNet, ProtT5 (Elnaggar et al., 2020), CPCProt (Lu et al., 2020), PLUS-RNN (Min et al., 2020). Via the "*extract*" stage, the pipeline incorporates supervised and unsupervised approaches for protein embeddings to further enhance analytical potential out-of-the-box. For instance, users can extract secondary structure in 3- and 8-states for embeddings from SeqVec (Heinzinger et al., 2019) and ProtBert (Elnaggar et al., 2020), or transfer GO annotations using embeddings of any available LM (Littmann et al., 2021). Pipeline runs are reproducible, as configurations are defined through files, and the output is stored in easily exchangeable formats, e.g., CSVs, FASTA, and HDF5 (The HDF Group, 2000).

For researchers contributing new protein LMs, *bio_embeddings* can provide a unified interface to distribute their work to the community, requiring minimal changes for pipeline consumers to make use of new protein LMs. For researchers contributing downstream uses of protein LMs [e.g., for the visualization of attention maps (Vig et al., 2020), which are most closely related to protein contact maps, or for the alignment of protein sequences (Morton et al., 2020)], the *bio_embeddings* pipeline provides a flexible approach to incorporate their work and directly extends it to all the LMs supported by *bio_embeddings*. In the future, as we expect more protein LMs to be developed, the *bio_embeddings* pipeline could be combined with the TAPE (Rao et al., 2019) evaluation system to provide an intuition for protein LM researchers about the best use of their new representations.

Critical Parameters

We strongly encourage users interested in generating their own sequence embeddings to do so on GPU-equipped machines, where the GPUs have at least 4 GB of vRAM and support CUDA® 11.0. While it is possible to generate embeddings via CPU computing, the slowdown with respect to GPU computing is significant and prohibitive for large sequence sets.

Differences in LM choice, sequence sets or parameters (e.g., UMAP) may lead to significantly different results than discussed in the protocols. While trying out the above steps on your own datasets is the ultimate goal, we encourage users to first try to execute the steps as laid out above to get a sense of the baseline behavior.

Troubleshooting

If you experience issues when installing the *bio_embedding* package, or when executing the steps laid out above, you may want to try to restart the Google Colab, or, if you are running the code locally, create a new python environment [e.g., by using Anaconda ("Anaconda Software Distribution," 2020)]. In our experience, the most common issues are caused by installation problems, or limited computational resources. To address the former, you might want to consider using docker instead of python (this is available at the source code, see "Internet Resources"). To address the latter, you might want to discuss solutions with your local research computing facilities or try an online service (see "Internet Resources").

Understanding Results

Basic Protocol 1

Through the steps outlined in this protocol, you generated an interactive plot of about 100 protein sequences with annotations of disorder content (either presenting high or low disorder content).

Basic Protocol 2

Through the steps outlined in this protocol, you generated embeddings for amino acids in sequences (`embeddings_file.h5`) and for sequences (`reduced_embeddings_file.h5`) from the DeepLoc sequence set. These files can be used on per-residue tasks (e.g., predict secondary structure) or per-protein tasks (e.g., predict subcellular location).

Basic Protocol 3

Through the steps outlined in this protocol, you generated interactive plots of sequence embeddings. You used color in plots to highlight annotated subcellular localization (from DeepLoc), and could test out different parameter choices (via Alternate Protocol 1) and annotations (via Alternate Protocol 2). You learned how to incorporate these steps in a *bio_embeddings* pipeline file to enable other researchers to reproduce your results (via the Support Protocol).

Basic Protocol 4

Through the steps outlined in this protocol, you trained a neural network on embeddings to predict subcellular localization of sequence embeddings.

Time Considerations

Basic Protocol 1

On a 2016 MacBook Pro with 16 GB of RAM, executing the pipeline took approximately 3 min. Considering installation of required software and download of necessary files, the overall execution time of the protocol should not exceed 20 min.

Basic Protocol 2

On an Nvidia 1080 GPU equipped with 8 GB of vRAM, embedding the whole DeepLoc dataset took ∼30 min. On a CPU (Intel i7-6700, 64 GB system RAM), embedding the sampled DeepLoc set took ∼2 min, while embedding the whole set took approximately 8 hr and 40 min. Executing the steps, not considering computation time, may take up to 30 min.

Basic Protocol 3

On Google Colab, the UMAP projection step (the most computationally expensive step) takes about 10 min. Writing the code and executing the steps, considering computation time, may take up to 1 hr.

Basic Protocol 4

On Google Colab, training various classifiers via grid search (the most computationally expensive step) takes about 15 min. Writing the code and executing the steps, considering computation time, may take up to 1 hr.

Acknowledgments

The authors thank Tim Karl (TUM) for help with hardware and software and Inga Weise (TUM) for support with many other aspects of this work. The authors thank Tom Sercu, Ali Madani, Daniel Berenberg, Alex Rives, Vladimir Gligorijevic, and Josh Meier for constructive discussions around protein language models and their use. The authors thank Roshan Rao, Neil Thomas, and Nicholas Bhattacharya for creating and maintaining TAPE. The authors also thank all those who deposited their experimental data in public databases, and those who maintain these databases. In particular, the authors thank Ioanis Xenarios (SIB, Univ. Lausanne), Matthias Uhlen (Univ. Upssala), and their teams at SwissProt and HPA. This work was supported by the Deutsche Forschungsgemeinschaft (DFG), project number RO1320/4-1, by the Bundesministerium für Bildung und Forschung (BMBF), project number 031L0168, and by the BMBF through the program "Software Campus 2.0 (TU München)", project number 01IS17049.

Open access funding enabled and organized by Projekt DEAL.

Author Contributions

Christian Dallago: Conceptualization, Data curation, Funding acquisition, Methodology, Project administration, Resources, Software, Supervision, Visualization, Writing-original draft, Writing-review & editing, Konstantin Schütze: Methodology, Software, Writing-review & editing, Michael Heinzinger: Conceptualization, Investigation, Software, Writing-review & editing, Tobias Olenyi: Software, Writing-review & editing, Maria Littmann: Writing-original draft, Writing-review & editing, Amy X. Lu: Writing-original draft, Writing-review & editing, Kevin K. Yang, Seonwoo Min: Writing-original draft, Writing-review & editing, Sungroh Yoon: Writing-original draft, James T. Morton: Writing-original draft, Writing-review & editing, Burkhard Rost: Conceptualization, Funding acquisition, Supervision, Writing-original draft, Writing-review & editing

Conflicts of Interest

A.L. is employed at Insitro, South San Francisco, CA, 94080. Insitro had no involvement in the design or implementation of the work presented here.

Data Availability Statement

The data that support the presented protocols are available at: *https://github.com/sacdallago/bio_embeddings*. These data were derived from the following resources available in the public domain: DisProt (https://www.disprot.org), DeepLoc (*http://www.cbs.dtu.dk/services/DeepLoc*).

Literature Cited

Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M., & Church, G. M. (2019). Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods*, *16*(12), 1315–1322. doi: 10.1038/s41592-019-0598-1.

Almagro Armenteros, J. J., Sønderby, C. K., Sønderby, S. K., Nielsen, H., & Winther, O. (2017). DeepLoc: Prediction of protein subcellular localization using deep learning. *Bioinformatics*, *33*(21), 3387–3395. doi: 10.1093/bioinformatics/btx431.

AlQuraishi, M. (2019). End-to-end differentiable learning of protein structure. *Cell Systems*, *8*(4), 292–301.e3. doi: 10.1016/j.cels.2019.03.006.

Anaconda Software Distribution. (2020). In *Anaconda Documentation* (Vers. 2-2.4.0) [Computer software]. Anaconda Inc. Available at https://docs.anaconda.com/.

Armenteros, J. J. A., Johansen, A. R., Winther, O., & Nielsen, H. (2020). Language modelling for biological sequences—curated datasets and baselines. *BioRxiv*, 2020.03.09.983585. doi: 10.1101/2020.03.09.983585.

Bender, E. M., & Koller, A. (2020). Climbing towards NLU: On meaning, form, and understanding in the age of data. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 5185–5198. doi: 10.18653/v1/2020.acl-main.463.

Biasini, M., Bienert, S., Waterhouse, A., Arnold, K., Studer, G., Schmidt, T., … Schwede, T. (2014). SWISS-MODEL: Modelling protein tertiary and quaternary structure using evolutionary information. *Nucleic Acids Research*, *42*, W252–W288. doi: 10.1093/nar/gku340.

Bisong, E. (2019). Google colaboratory. In *Building machine learning and deep learning models on google cloud platform* (pp. 59–64). New York: Springer.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *ArXiv*, 1607.04606 [Cs]. Available at http://arxiv.org/abs/1607.04606.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., … Amodei, D. (2020). Language models are few-shot learners. *ArXiv*, 2005.14165 [Cs]. Available at http://arxiv.org/abs/2005.14165.

Callaway, E. (2020). 'It will change everything': DeepMind's AI makes gigantic leap in solving protein structures. *Nature*, *588*(7837), 203–204. doi: 10.1038/d41586-020-03348-4.

Chelba, C., Mikolov, T., Schuster, M., Ge, Q., Brants, T., Koehn, P., & Robinson, T. (2014). One billion word benchmark for measuring progress in statistical language modeling. *ArXiv*, 1312.3005 [Cs]. Available at http://arxiv.org/abs/1312.3005.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, 1810.04805 [Cs]. Available at http://arxiv.org/abs/1810.04805.

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., … Rost, B. (2020). ProtTrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing. In *BioRxiv*, 2020.07.12.199554. doi: 10.1101/2020.07.12.199554.

Goldberg, T., Hecht, M., Hamp, T., Karl, T., Yachdav, G., Ahmed, N., … others (2014). LocTree3 prediction of localization. *Nucleic Acids Research*, *42*(W1), W350–W355.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. doi: 10.1038/s41586-020-2649-2.

Hatos, A., Hajdu-Soltész, B., Monzon, A. M., Palopoli, N., Álvarez, L., Aykac-Fas, B., … Piovesan, D. (2020). DisProt: Intrinsic protein disorder annotation in 2020. *Nucleic Acids Research*, *48*(D1), D269–D276. doi: 10.1093/nar/gkz975.

The HDF Group. (2000, 2010). *Hierarchical data format version 5*. Available at http://www.hdfgroup.org/HDF5.

Heinzerling, B. (2020). NLP's clever Hans moment has arrived. *Journal of Cognitive Science*, *21*(1), 159–167.

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. In *BMC Bioinformatics*, *20*, 723. doi: 10.1186/s12859-019-3220-8.

Henikoff, S., & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, *89*(22), 10915–10919. doi: 10.1073/pnas.89.22.10915.

Hopf, T. A., Colwell, L. J., Sheridan, R., Rost, B., Sander, C., & Marks, D. S. (2012). Three-dimensional structures of membrane proteins from genomic sequencing. *Cell*, *149*(7), 1607–1621. doi: 10.1016/j.cell.2012.04.012.

Hopf, T. A., Ingraham, J. B., Poelwijk, F. J., Schärfe, C. P., Springer, M., Sander, C., & Marks, D. S. (2017). Mutation effects predicted from sequence co-variation. *Nature Biotechnology*, *35*(2), 128–135.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., … others (2016). Jupyter Notebooks-a publishing format for reproducible computational workflows. *ELPUB*, 87–90.

Littmann, M., Heinzinger, M., Dallago, C., Olenyi, T., & Rost, B. (2021). Embeddings from deep learning transfer GO annotations beyond homology. *Scientific Reports*, *11*(1), 1160. doi: 10.1038/s41598-020-80786-0.

Lu, A. X., Zhang, H., Ghassemi, M., & Moses, A. (2020). Self-supervised contrastive learning of protein representations by mutual information maximization. *BioRxiv*, 2020.09.04.283929. doi: 10.1101/2020.09.04.283929.

Madani, A., McCann, B., Naik, N., Keskar, N. S., Anand, N., Eguchi, R. R., … Socher, R. (2020). ProGen: Language modeling for protein generation. *BioRxiv*, 2020.03.07.982272. doi: 10.1101/2020.03.07.982272.

Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In A. F. Gelbukh (Ed.), *Computational Linguistics and Intelligent Text Processing* (pp. 171–189). New York: Springer. doi: 10.1007/978-3-642-19400-9_14.

McCoy, T., Pavlick, E., & Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3428–3448. doi: 10.18653/v1/P19-1334.

McInnes, L., Healy, J., & Melville, J. (2018). UMAP: Uniform manifold approximation and projection for dimension reduction. *ArXiv*, 1802.03426 [Cs, Stat]. Available at http://arxiv.org/abs/1802.03426.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *ArXiv*, 1301.3781 [Cs]. Available at http://arxiv.org/abs/1301.3781.

Min, S., Park, S., Kim, S., Choi, H.-S., & Yoon, S. (2020). Pre-training of deep bidirectional protein sequence representations with structural information. *ArXiv*, 1912.05625 [Cs, q-Bio, Stat]. Available at http://arxiv.org/abs/1912.05625.

Morcos, F., Pagnani, A., Lunt, B., Bertolino, A., Marks, D. S., Sander, C., … Weigt, M. (2011). Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, *108*(49), E1293–E1301. doi: 10.1073/pnas.1111471108.

Morton, J. T., Strauss, C. E. M., Blackwell, R., Berenberg, D., Gligorijevic, V., & Bonneau, R. (2020). Protein structural alignments from sequence. *BioRxiv*, 2020.11.03.365932. doi: 10.1101/2020.11.03.365932.

Moult, J., Pedersen, J. T., Judson, R., & Fidelis, K. (1995). A large-scale experiment to assess protein structure prediction methods. *Proteins*, *23*, ii–iv.

Ovchinnikov, S., Park, H., Varghese, N., Huang, P.-S., Pavlopoulos, G. A., Kim, D. E., … Baker, D. (2017). Protein structure determination using metagenome sequence data. *Science*, *355*(6322), 294–298. doi: 10.1126/science.aah4043.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. October 25-29, 2014, Doha, Qatar.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *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. June 1-June 6, 2018, New Orleans, Louisiana. doi: 10.18653/v1/N18-1202.

Pires, T., Schlinger, E., & Garrette, D. (2019). How multilingual is multilingual BERT? *ArXiv*, 1906.01502 [Cs]. Available at http://arxiv.org/abs/1906.01502.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., … Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv*, 1910.10683 [Cs, Stat]. Available at http://arxiv.org/abs/1910.10683.

Raina, R., Battle, A., Lee, H., Packer, B., & Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. *Proceedings of the 24th International Conference on Machine Learning*, 759–766. Bellevue, Washington. doi: 10.1145/1273496.1273592.

Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, P., Canny, J., … Song, Y. (2019). Evaluating Protein Transfer Learning with TAPE. In H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems 32* (pp. 9689–9701). Curran Associates, Inc. Available at http://papers.nips.cc/paper/9163-evaluating-protein-transfer-learning-with-tape.pdf.

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

Reeb, J., Goldberg, T., Ofran, Y., & Rost, B. (2020). Predictive methods using protein sequences. In A. D. Baxevanis, G. D. Bader, & D. S. Wishart (Eds.) *Bioinformatics* (4th ed., p. 185).

Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C. L., … Fergus, R. (2019). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *BioRxiv*, 622803. doi: 10.1101/622803.

Rost, B. (1996). PHD: Predicting one-dimensional protein structure by profile based neural networks. *Methods in Enzymology*, *266*, 525–539.

Rost, B. (2001). Protein secondary structure prediction continues to rise. *Journal of Structural Biology*, *134*, 204–218.

Rost, B., & Sander, C. (1993). Prediction of protein secondary structure at better than 70% accuracy. *Journal of Molecular Biology*, *232*, 584–599.

Rost, B., & Sander, C. (1994). Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins*, *19*, 55–72.

Rost, B., & Sander, C. (1995). Progress of 1D protein structure prediction at last. *Proteins*, *23*, 295–300.

Saputro, D. R. S., & Widyaningsih, P. (2017). Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) method for the parameter estimation on geographically weighted ordinal logistic regression model (GWOLR). *AIP Conference Proceedings*, *1868*(1), 040009. doi: 10.1063/1.4995124.

Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., … Carin, L. (2018). Baseline

needs more love: On simple word-embedding-based models and associated pooling mechanisms. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 440–450. July 15 - 20, 2018, Melbourne, Australia. doi: 10.18653/v1/P18-1041.

Steinegger, M., Meier, M., Mirdita, M., Vöhringer, H., Haunsberger, S. J., & Söding, J. (2019). HH-suite3 for fast remote homology detection and deep protein annotation. *BMC Bioinformatics*, *20*(1), 473. doi: 10.1186/s12859-019-3019-7.

Steinegger, M., Mirdita, M., & Söding, J. (2019). Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nature Methods*, *16*(7), 603–606. doi: 10.1038/s41592-019-0437-4.

Steinegger, M., & Söding, J. (2018). Clustering huge protein sequence sets in linear time. *Nature Communications*, *9*(1), 2542. doi: 10.1038/s41467-018-04964-5.

Stormo, G. D., Schneider, T. D., Gold, L., & Ehrenfeucht, A. (1982). Use of the 'Perceptron' algorithm to distinguish translational initiation sites in *E. coli*. *Nucleic Acids Research*, *10*(9), 2997–3011. doi: 10.1093/nar/10.9.2997.

The UniProt Consortium. (2019). UniProt: A worldwide hub of protein knowledge. *Nucleic Acids Research*, *47*(D1), D506–D515. doi: 10.1093/nar/gky1049.

Vig, J., Madani, A., Varshney, L. R., Xiong, C., Socher, R., & Rajani, N. F. (2020). BERTology meets biology: Interpreting attention in protein language models. *ArXiv*, 2006.15222 [Cs, q-Bio]. Available at http://arxiv.org/abs/2006.15222.

Villegas-Morcillo, A., Makrodimitris, S., van Ham, R. C. H. J., Gomez, A. M., Sanchez, V., & Reinders, M. J. T. (2020). Unsupervised protein embeddings outperform hand-crafted sequence and structure features at predicting molecular function. *Bioinformatics*, *2020*, btaa701. doi: 10.1093/bioinformatics/btaa701.

Zhu, J., Xia, Y., Wu, L., He, D., Qin, T., Zhou, W., … Liu, T.-Y. (2020). Incorporating BERT into neural machine translation. *ArXiv*, 2002.06823 [Cs]. Available at http://arxiv.org/abs/2002.06823.

Internet Resources

https://github.com/sacdallago/bio_embeddings.

bio_embeddings source code.

https://docs.bioembeddings.com.

bio_embeddings Python documentation

http://examples.bioembeddings.com.

Example bio_embeddings pipeline runs.

http://notebooks.bioembeddings.com.

Notebooks for interactive bio-embeddings workflows.

For small FASTA files (<20000 residues in total) it is also possible to use the bio_embeddings web pipeline: https://api.bioembeddings.com. The web pipeline also allows execution of single sequences (<2000 residues) instantaneously, as utilized by PredictProtein (https://predictprotein.org) and https://embed.protein.properties.

JOHN WILEY AND SONS LICENSE
TERMS AND CONDITIONS

May 23, 2022

This Agreement between Mr. Christian Dallago ("You") and John Wiley and Sons ("John Wiley and Sons") consists of your license details and the terms and conditions provided by John Wiley and Sons and Copyright Clearance Center.

License Number	5314710554640
License date	May 23, 2022
Licensed Content Publisher	John Wiley and Sons
Licensed Content Publication	Current Protocols
Licensed Content Title	Learned Embeddings from Deep Learning to Visualize and Predict Protein Sets
Licensed Content Author	Burkhard Rost, James T. Morton, Sungroh Yoon, et al
Licensed Content Date	May 7, 2021
Licensed Content Pages	1
Type of use	Dissertation/Thesis
Requestor type	Author of this Wiley article
Format	Print and electronic
Portion	Full article
Will you be translating?	No

Title	Biases Model Machine Learning Predictions in Protein Biology
Institution name	Technical University of Munich
Expected presentation date	Oct 2022
Order reference number	cpz1.113
Requestor Location	Mr. Christian Dallago bolzmanstrasse 3 Garching bei muenchen, bayern 85748 Germany Attn: Mr. Christian Dallago
Publisher Tax ID	EU826007151
Total	0.00 USD

Terms and Conditions

TERMS AND CONDITIONS

This copyrighted material is owned by or exclusively licensed to John Wiley & Sons, Inc. or one of its group companies (each a"Wiley Company") or handled on behalf of a society with which a Wiley Company has exclusive publishing rights in relation to a particular work (collectively "WILEY"). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the billing and payment terms and conditions established by the Copyright Clearance Center Inc., ("CCC's Billing and Payment terms and conditions"), at the time that you opened your RightsLink account (these are available at any time at http://myaccount.copyright.com).

Terms and Conditions

- The materials you have requested permission to reproduce or reuse (the "Wiley Materials") are protected by copyright.
- You are hereby granted a personal, non-exclusive, non-sub licensable (on a stand-alone basis), non-transferable, worldwide, limited license to reproduce the Wiley Materials for the purpose specified in the licensing process. This license, **and any CONTENT (PDF or image file) purchased as part of your order,** is for a one-time use only and limited to any maximum distribution number specified in the license. The first instance of republication or reuse granted by this license must be completed within two years of the date of the grant of this license (although copies prepared before the end date may be distributed thereafter). The Wiley Materials shall not be used in any other manner or for any other purpose, beyond what is granted in the

license. Permission is granted subject to an appropriate acknowledgement given to the author, title of the material/book/journal and the publisher. You shall also duplicate the copyright notice that appears in the Wiley publication in your use of the Wiley Material. Permission is also granted on the understanding that nowhere in the text is a previously published source acknowledged for all or part of this Wiley Material. Any third party content is expressly excluded from this permission.

- With respect to the Wiley Materials, all rights are reserved. Except as expressly granted by the terms of the license, no part of the Wiley Materials may be copied, modified, adapted (except for minor reformatting required by the new Publication), translated, reproduced, transferred or distributed, in any form or by any means, and no derivative works may be made based on the Wiley Materials without the prior permission of the respective copyright owner.**For STM Signatory Publishers clearing permission under the terms of the STM Permissions Guidelines only, the terms of the license are extended to include subsequent editions and for editions in other languages, provided such editions are for the work as a whole in situ and does not involve the separate exploitation of the permitted figures or extracts,** You may not alter, remove or suppress in any manner any copyright, trademark or other notices displayed by the Wiley Materials. You may not license, rent, sell, loan, lease, pledge, offer as security, transfer or assign the Wiley Materials on a stand-alone basis, or any of the rights granted to you hereunder to any other person.

- The Wiley Materials and all of the intellectual property rights therein shall at all times remain the exclusive property of John Wiley & Sons Inc, the Wiley Companies, or their respective licensors, and your interest therein is only that of having possession of and the right to reproduce the Wiley Materials pursuant to Section 2 herein during the continuance of this Agreement. You agree that you own no right, title or interest in or to the Wiley Materials or any of the intellectual property rights therein. You shall have no rights hereunder other than the license as provided for above in Section 2. No right, license or interest to any trademark, trade name, service mark or other branding ("Marks") of WILEY or its licensors is granted hereunder, and you agree that you shall not assert any such right, license or interest with respect thereto

- NEITHER WILEY NOR ITS LICENSORS MAKES ANY WARRANTY OR REPRESENTATION OF ANY KIND TO YOU OR ANY THIRD PARTY, EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO THE MATERIALS OR THE ACCURACY OF ANY INFORMATION CONTAINED IN THE MATERIALS, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF MERCHANTABILITY, ACCURACY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, USABILITY, INTEGRATION OR NON-INFRINGEMENT AND ALL SUCH WARRANTIES ARE HEREBY EXCLUDED BY WILEY AND ITS LICENSORS AND WAIVED BY YOU.

- WILEY shall have the right to terminate this Agreement immediately upon breach of this Agreement by you.

- You shall indemnify, defend and hold harmless WILEY, its Licensors and their respective directors, officers, agents and employees, from and against any actual or threatened claims, demands, causes of action or proceedings arising from any breach of this Agreement by you.

- IN NO EVENT SHALL WILEY OR ITS LICENSORS BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR ENTITY FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, INDIRECT, EXEMPLARY OR PUNITIVE DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, PROVISIONING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION,

WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

- Should any provision of this Agreement be held by a court of competent jurisdiction to be illegal, invalid, or unenforceable, that provision shall be deemed amended to achieve as nearly as possible the same economic effect as the original provision, and the legality, validity and enforceability of the remaining provisions of this Agreement shall not be affected or impaired thereby.

- The failure of either party to enforce any term or condition of this Agreement shall not constitute a waiver of either party's right to enforce each and every term and condition of this Agreement. No breach under this agreement shall be deemed waived or excused by either party unless such waiver or consent is in writing signed by the party granting such waiver or consent. The waiver by or consent of a party to a breach of any provision of this Agreement shall not operate or be construed as a waiver of or consent to any other or subsequent breach by such other party.

- This Agreement may not be assigned (including by operation of law or otherwise) by you without WILEY's prior written consent.

- Any fee required for this permission shall be non-refundable after thirty (30) days from receipt by the CCC.

- These terms and conditions together with CCC's Billing and Payment terms and conditions (which are incorporated herein) form the entire agreement between you and WILEY concerning this licensing transaction and (in the absence of fraud) supersedes all prior agreements and representations of the parties, oral or written. This Agreement may not be amended except in writing signed by both parties. This Agreement shall be binding upon and inure to the benefit of the parties' successors, legal representatives, and authorized assigns.

- In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall prevail.

- WILEY expressly reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

- This Agreement will be void if the Type of Use, Format, Circulation, or Requestor Type was misrepresented during the licensing process.

- This Agreement shall be governed by and construed in accordance with the laws of the State of New York, USA, without regards to such state's conflict of law rules. Any legal action, suit or proceeding arising out of or relating to these Terms and Conditions or the breach thereof shall be instituted in a court of competent jurisdiction in New York County in the State of New York in the United States of America and each party hereby consents and submits to the personal jurisdiction of such court, waives any objection to venue in such court and consents to service of process by registered or certified mail, return receipt requested, at the last known address of such party.

WILEY OPEN ACCESS TERMS AND CONDITIONS

Wiley Publishes Open Access Articles in fully Open Access Journals and in Subscription journals offering Online Open. Although most of the fully Open Access journals publish open access articles under the terms of the Creative Commons Attribution (CC BY) License only, the subscription journals and a few of the Open Access Journals offer a choice of Creative Commons Licenses. The license type is clearly identified on the article.

The Creative Commons Attribution License

The Creative Commons Attribution License (CC-BY) allows users to copy, distribute and transmit an article, adapt the article and make commercial use of the article. The CC-BY license permits commercial and non-

Creative Commons Attribution Non-Commercial License

The Creative Commons Attribution Non-Commercial (CC-BY-NC)License permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.(see below)

Creative Commons Attribution-Non-Commercial-NoDerivs License

The Creative Commons Attribution Non-Commercial-NoDerivs License (CC-BY-NC-ND) permits use, distribution and reproduction in any medium, provided the original work is properly cited, is not used for commercial purposes and no modifications or adaptations are made. (see below)

Use by commercial "for-profit" organizations

Use of Wiley Open Access articles for commercial, promotional, or marketing purposes requires further explicit permission from Wiley and will be subject to a fee.

Further details can be found on Wiley Online Library
http://olabout.wiley.com/WileyCDA/Section/id-410895.html

Other Terms and Conditions:

v1.10 Last updated September 2015

Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.

5.3 PREDICTPROTEIN-PREDICTING PROTEIN STRUCTURE AND FUNCTION FOR 29 YEARS

Summary. Since its 1992 launch, PredictProtein (https://predictprotein.org/) has been a one-stop online resource for protein analysis. In 2020, for an average of 3000 monthly users, PredictProtein combined over 13 tools into a single resource. From just an input protein sequence, the server provides online visualizations of multiple sequence alignments (MSAs), predictions of protein structure (secondary structure, solvent accessibility, transmembrane segments, disordered regions, protein flexibility, and disulfide bridges) and function (variant effect, GO terms, subcellular localization, and protein-, RNA-, and DNA binding sites). By additionally providing computable artifacts (via programmatic access), the server caters the needs of computational and experimental biologists alike. Offline use of PredictProtein tools is enabled via an omni-docker container: quickly installed on single machines and clusters. Since the previous major update in 2014, PredictProtein's infrastructure was enhanced to offer more reliable execution, more storage space and decreased runtime for predictions. Runtime was also cut four-fold by sourcing alignment generation to MMseqs2 (M Mirdita *et al.*, 2021). Usability was improved via new UI elements (Watkins *et al.*, 2017). Prediction methods for DNA-, RNA- and protein binding and GO annotations have been replaced with revised methods (Qiu *et al.*, 2020; Littmann, Heinzinger, Dallago, Olenyi, *et al.*, 2021). ProtT5-sec, an alternative secondary structure prediction method based on cutting-edge Deep Learning techniques (Elnaggar *et al.*, 2021), was integrated side-by-side to evolution-based RePROF. The PredictProtein server offers access to a vast range of accurate predictors, many topping the leaderboards even after a decade, with new recently integrated methods to boost the breadth of available sequence features and improve accuracy on dated methods.

Relevance. PredictProtein has served users with predictions of protein properties for almost 30 years. Its relevance to the field and to this thesis are manyfold: from providing landmark solutions to characterize proteins, pushing the boundaries of "known" sequence space, to integrating intuitive visualizations to simplify interpretation of complex machine learning predictions for non-experts. The most significant scientific update in the 2021 edition of PredictProtein was the integration of cutting edge pLM models, on the one hand cementing the foundation to their use for the broader community, on the other hand, signaling a shift in how computational predictions of proteins are used. While previous models focused mainly on predicting attributes of proteins, e.g., subcellular localization, through embeddings computational biologists can access the underlying representation of proteins, enabling custom analyses of proteins from a high dimensional embedding without categorization into narrow, supervised ontologies.

Contribution. I am one of four principal authors of this paper. I am also the corresponding author. I contributed conceptualization and writing.

Copyright notice. The original publication is available in open access at the DOI 10.1093/nar/gkab354 and in the following. The copyright notice is attached in this appendix following the manuscript.

Published online 17 May 2021 Nucleic Acids Research, 2021, Vol. 49, Web Server issue W535–W540
https://doi.org/10.1093/nar/gkab354

PredictProtein - Predicting Protein Structure and Function for 29 Years

Michael Bernhofer[1,2,†], Christian Dallago[1,2,*,†], Tim Karl[1,†], Venkata Satagopam[3,4,†], Michael Heinzinger[1,2], Maria Littmann[1,2], Tobias Olenyi[1], Jiajun Qiu[1,5], Konstantin Schütze[1], Guy Yachdav[1], Haim Ashkenazy[6,7], Nir Ben-Tal[8], Yana Bromberg[9], Tatyana Goldberg[1], Laszlo Kajan[10], Sean O'Donoghue[11], Chris Sander[12,13,14], Andrea Schafferhans[1,15], Avner Schlessinger[16], Gerrit Vriend[17], Milot Mirdita[18], Piotr Gawron[3], Wei Gu[3,4], Yohan Jarosz[3,4], Christophe Trefois[3,4], Martin Steinegger[19,20], Reinhard Schneider[3,4] and Burkhard Rost[1,21,22,*]

[1]TUM (Technical University of Munich) Department of Informatics, Bioinformatics & Computational Biology - i12, Boltzmannstr 3, 85748 Garching/Munich, Germany, [2]TUM Graduate School CeDoSIA, Boltzmannstr 11, 85748 Garching, Germany, [3]Luxembourg Centre For Systems Biomedicine (LCSB), University of Luxembourg, Campus Belval, House of Biomedicine II, 6 avenue du Swing, L-4367 Belvaux, Luxembourg, [4]ELIXIR Luxembourg (ELIXIR-LU) Node, University of Luxembourg, Campus Belval, House of Biomedicine II, 6 avenue du Swing, L-4367 Belvaux, Luxembourg, [5]Department of Otolaryngology Head & Neck Surgery, The Ninth People's Hospital & Ear Institute, School of Medicine & Shanghai Key Laboratory of Translational Medicine on Ear and Nose Diseases, Shanghai Jiao Tong University, Shanghai, China, [6]Department of Molecular Biology, Max Planck Institute for Developmental Biology, Tübingen, Germany, [7]The Shmunis School of Biomedicine and Cancer Research, George S. Wise Faculty of Life Sciences, Tel Aviv University, 69978 Tel Aviv, Israel, [8]Department of Biochemistry & Molecular Biology, George S. Wise Faculty of Life Sciences, Tel Aviv University, 69978 Tel Aviv, Israel, [9]Department of Biochemistry and Microbiology, Rutgers University, New Brunswick, NJ 08901, USA, [10]Roche Polska Sp. z o.o., Domaniewska 39B, 02–672 Warsaw, Poland, [11]Garvan Institute of Medical Research, Sydney, Australia, [12]Department of Data Sciences, Dana-Farber Cancer Institute, Boston, MA 02215, USA, [13]Department of Cell Biology, Harvard Medical School, Boston, MA 02215, USA, [14]Broad Institute of MIT and Harvard, Boston, MA 02142, USA, [15]HSWT (Hochschule Weihenstephan Triesdorf | University of Applied Sciences), Department of Bioengineering Sciences, Am Hofgarten 10, 85354 Freising, Germany, [16]Department of Pharmacological Sciences, Icahn School of Medicine at Mount Sinai, New York, NY 10029, USA, [17]BIPS, Poblacion Baco, Mindoro, Philippines, [18]Quantitative and Computational Biology, Max Planck Institute for Biophysical Chemistry, Göttingen, Germany, [19]School of Biological Sciences, Seoul National University, Seoul, South Korea, [20]Artificial Intelligence Institute, Seoul National University, Seoul, South Korea, [21]Institute for Advanced Study (TUM-IAS), Lichtenbergstr. 2a, 85748 Garching/Munich, Germany and [22]TUM School of Life Sciences Weihenstephan (WZW), Alte Akademie 8, Freising, Germany

Received February 23, 2021; Revised April 06, 2021; Editorial Decision April 21, 2021; Accepted May 10, 2021

ABSTRACT

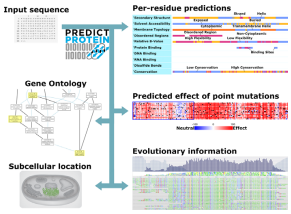
Since 1992 *PredictProtein* (https://predictprotein.org) is a one-stop online resource for protein sequence analysis with its main site hosted at the Luxembourg Centre for Systems Biomedicine (LCSB) and queried monthly by over 3,000 users in 2020. *PredictProtein* was the first Internet server for protein predictions. It pioneered combining evolutionary information and machine learning. Given a protein sequence as input, the server outputs multiple sequence alignments, predictions of protein structure in 1D and 2D (secondary structure, solvent accessibility, transmembrane segments, disordered regions, protein flexibility, and disulfide bridges) and predictions of protein function (functional effects of sequence variation or point mutations, Gene Ontology (GO) terms, subcellular localization, and

*To whom correspondence should be addressed. Tel: +49 289 17 811; Email: christian.dallago@tum.de
Correspondence may also be addressed to Burkhard Rost. Email: assistant@rostlab.org
†The authors wish it to be known that, in their opinion, the first four authors should be regarded as joint First Authors.

W536 *Nucleic Acids Research, 2021, Vol. 49, Web Server issue*

protein-, RNA-, and DNA binding). *PredictProtein's* infrastructure has moved to the LCSB increasing throughput; the use of MMseqs2 sequence search reduced runtime five-fold (apparently without lowering performance of prediction methods); user interface elements improved usability, and new prediction methods were added. *PredictProtein* recently included predictions from deep learning embeddings (GO and secondary structure) and a method for the prediction of proteins and residues binding DNA, RNA, or other proteins. PredictProtein.org aspires to provide reliable predictions to computational and experimental biologists alike. All scripts and methods are freely available for offline execution in high-throughput settings.

GRAPHICAL ABSTRACT



INTRODUCTION

The sequence is known for far more proteins (1) than experimental annotations of function or structure (2,3). This sequence-annotation gap existed when *PredictProtein* (4,5) started in 1992, and has kept expanding ever since (6). Unannotated sequences contribute crucial evolutionary information to neural networks predicting secondary structure (7,8) that seeded *PredictProtein (PP)* at the European Molecular Biology Laboratory (EMBL) in 1992 (9), the first fully automated, query-driven Internet server providing evolutionary information and structure prediction for any protein. Many other methods predicting aspects of protein function and structure have since joined under the PP roof (4,5,10) now hosted by the Luxembourg Centre of Systems Biomedicine (LCSB).

PP offers an array of structure and function predictions most of which combine machine learning with evolutionary information; now enhanced by a faster alignment algorithm (11,12). A few prediction methods now also use embeddings (13,14) from protein Language Models (LMs) (13–18). Embeddings are much faster to obtain than evolutionary information, yet for many tasks, perform almost as well, or even better (19,20). All PP methods join at PredictProtein.org with interactive visualizations; for some methods, more advanced visualizations are linked (21–23). The reliability of *PredictProtein*, its speed, the continuous integration of well-validated, top methods, and its intuitive interface have attracted thousands of researchers over 29 years of steady operation.

MATERIALS AND METHODS

PredictProtein (PP) provides

multiple sequence alignments (MSAs) and position-specific scoring matrices (PSSMs) computed by a combination of pairwise BLAST (24), PSI-BLAST (25), and MMseqs2 (11,12) on query vs. PDB (26) and query versus UniProt (1). For each residue in the query, the following per-residue predictions are assembled: secondary structure (RePROF/PROFsec (5,27) and ProtBertSec (14)); solvent accessibility (RePROF/PROFacc); transmembrane helices and strands (TMSEG (28) and PROFtmb (29)); protein disorder (Meta-Disorder (30)); backbone flexibility (relative B-values; PROFbval (31)); disulfide bridges (DISULFIND (32)); sequence conservation (ConSurf/ConSeq (33–36)); protein-protein, protein-DNA, and protein-RNA binding residues (ProNA2020 (3)); PROSITE motifs (37); effects of sequence variation (single amino acid variants, SAVs; SNAP2 (38)). For each query per-protein predictions include: transmembrane topology (TMSEG (28)); binary protein-(DNA|RNA|protein) binding (protein binds X or not; ProNA2020 (3)); Gene Ontology (GO) term predictions (goPredSim (19)); subcellular localization (LocTree3 (39)); Pfam (40) domain scans, and some biophysical features. Under the hood, PP computes more results (SOM: PredictProtein Methods; Supplementary Table S1), either as input for frontend methods, or for legacy support.

New: goPredSim embedding-based transfer of Gene Ontology (GO)

goPredSim (19) predicts GO terms by transferring annotations from the most embedding-similar protein. Embeddings are obtained from SeqVec (13); similarity is established through the Euclidean distance between the embedding of a query and a protein with experimental GO annotations. Replicating the conditions of CAFA3 (41) in 2017, goPredSim achieved F_{max} values of 37 ± 2%, 52 ± 2% and 58 ± 2% for BPO (biological process), MFO (molecular function), and CCO (cellular component), respectively (41,42). Using Gene Ontology Annotation (GOA) (43,44) to test 296 proteins annotated after February 2020, goPredSim appeared to reach even slightly higher values that were confirmed by CAFA4 (45).

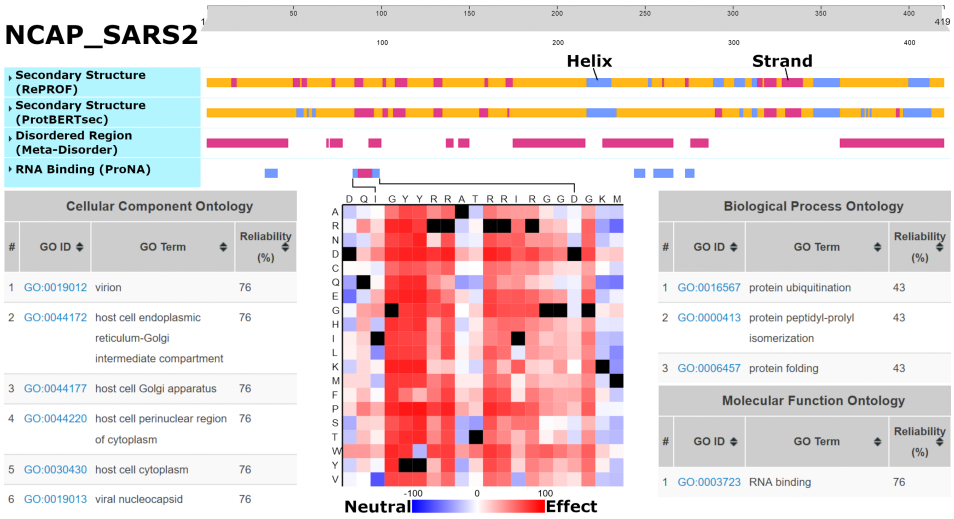
New: ProtBertSec secondary structure prediction

ProtBertSec predicts secondary structure in three states (helix, strand, other) using ProtBert (14) embeddings derived from training on BFD with almost 3×10^9 proteins (6,46). On a hold-out set from CASP12, ProtBertSec reached a three-state per-residue accuracy of Q3 = 76 ± 1.5% (47). Although below the state-of-the-art (NetSurfP-2.0 (48) at 82%), this method performed on-par with other MSA-based methods, despite itself not using MSAs.

New: ProNA2020 protein–protein, protein–RNA and protein–DNA

ProNA2020 (3) predicts whether or not a protein interacts with other proteins, RNA or DNA (binary), and if so, where

Nucleic Acids Research, 2021, Vol. 49, Web Server issue W537



Cellular Component Ontology

#	GO ID	GO Term	Reliability (%)
1	GO:0019012	virion	76
2	GO:0044172	host cell endoplasmic reticulum-Golgi intermediate compartment	76
3	GO:0044177	host cell Golgi apparatus	76
4	GO:0044220	host cell perinuclear region of cytoplasm	76
5	GO:0030430	host cell cytoplasm	76
6	GO:0019013	viral nucleocapsid	76

Biological Process Ontology

#	GO ID	GO Term	Reliability (%)
1	GO:0016567	protein ubiquitination	43
2	GO:0000413	protein peptidyl-prolyl isomerization	43
3	GO:0006457	protein folding	43

Molecular Function Ontology

#	GO ID	GO Term	Reliability (%)
1	GO:0003723	RNA binding	76

Figure 1. Predictions for SARS-CoV-2 Nucleoprotein (NCAP_SARS2). Underneath the interactive slider at the top: RePROF and ProtBertSec secondary structure (blue: helix; purple: strand; orange: other); Meta-Disorder intrinsically disordered regions (purple); ProNA2020 RNA-binding residues (low confidence: blue; medium confidence: purple). goPredSim transfers of GeneOntology (GO) terms based on embedding similarity (lower left: CCO; lower right: BPO & MFO). SNAP2 predicts the effect of point-mutations on function for the RNA-binding region from I84 to D98 (bottom-center; black: native residue). Link: predictprotein.org/visual_results?req_id=1nAmulUQY$FRPFaP8NTqLW9DzdlTG3B/.

it binds (which residues). The binary per-protein predictions rely on homology and machine learning models employing profile-kernel SVMs (49) and on embeddings from an *in-house* implementation of ProtVec (50). Per-residue predictions (where) use simple neural networks due to data shortage (51–53). ProNA2020 correctly predicted 77 ± 1% of the proteins binding DNA, RNA or protein. In proteins known to bind other proteins, RNA or DNA, ProNA2020 correctly predicted 69 ± 1%, 81 ± 1% and 80 ± 1% of binding residues, respectively.

New: MMseqs2 speedy evolutionary information

Most time-consuming for PP was the search for related proteins in ever growing databases. MMseqs2 (11) finds related sequences blazingly fast and seeds a PSI-BLAST search (25). The query sequence is sent to a dedicated MMseqs2 server that searches for hits against cluster representatives within the UniClust30 (54) and PDB (26) reduced to 70% pairwise percentage sequence identity (PIDE). All hits and their respective cluster members are turned into a MSA and filtered to the 3000 most diverse sequences.

WEB SERVER

Frontend and user interface (UI)

Users query PredictProtein.org by submitting a protein sequence. Results are available in seconds for sequences that had been submitted recently (cf. *PPcache* next section), or within up to 30 min if predictions are recomputed. Per-residue predictions are displayed online via ProtVista (55), which allows to zoom into any sequential protein region (Supplementary Figure S1), and are grouped by category (e.g. secondary structure), which can be expanded to display more detail (e.g. helix, strand, other). On the results page, links to visualize MSAs through *Alignment Viewer* (56) are available. More predictions can be accessed through a menu on the left, e.g. *Gene Ontology Terms*, *Effect of Point Mutations* and *Subcellular Localization*. Prediction views include references and details of outputs, as well as rich visualizations, e.g. GO trees for GO predictions and cell images with highlighted predicted locations for subcellular localization predictions (57).

PPcache, backend and programmatic access

The PP backend lives at LCSB, allowing for up to 48 parallel queries. Results are stored on disc in the *PPcache* (5). Users submitting sequences for which results were over the last two years obtain results immediately. Using the bio-embeddings pipeline (58), the PPcache is enriched by embeddings and embedding-based predictions on the fly. For all methods displayed on the frontend, JSON files compliant with *ProtVista* (55) are available via REST APIs (SOM: Programmatic access), and are in use by external services such as the protein 3D structure visualization suite *Aquaria* (21,23) and by *MolArt* (22).

PredictProtein is available for local use

All results displayed on and downloadable from PP are available through Docker (59) and as source code for local and cloud execution (available at github.com/rostlab).

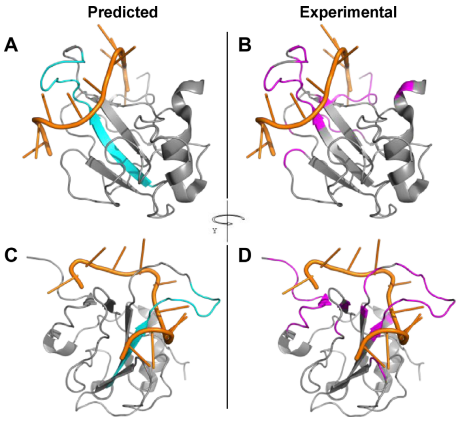


Figure 2. Experimental and predicted RNA-binding residues for NCAP2_SARS2. Predicted (via ProNA2020, in cyan, panels **A** and **C**) and observed (within 5Å, in magenta, panels **B** and **D**) RNA-binding residues for the SARS-CoV-2 nucleoprotein (gray) complexed with a 10-mer ssRNA (orange), PDB structure 7ACT (61). Two-third of the predictions are correct (precision = 0.73, recall = 0.20), which is around the expected average performance reported by ProNA2020. The important sequence consecutive central strand and loop are predicted well, while several short sequence segments that are far away in sequence space but close in structure space are missed, which is expected as ProNA2020 has no notion of 3D structure, i.e., cannot identify 'binding sites'. Panels A and B show a different orientation than panels C and D.

USE CASE

We demonstrate PredictProtein.org tools through predictions of the novel coronavirus SARS-CoV-2 (NCBI:txid2697049) nucleoprotein (UniProt identifier P0DTC9/NCAP_SARS2; Figure 1; SOM: Use Case; Supplementary Figure S2). NCAP_SARS2 has 419 residues and interacts with itself (experimentally verified). Sequence similarity and automatic assignment via UniRule (60) suggest NCAP is RNA-binding (binding with the viral genome), binding with the membrane protein M (UniProt identifier P0DTC5/VME1_SARS2), and is fundamental for virion assembly. goPredSim (19) transferred GO terms from other proteins for MFO (*RNA-binding*; GO:0003723; ECO:0000213) and CCO (compartments in the host cell and viral nucleocapsid; GO:0019013; GO:0044172; GO:0044177; GO:0044220; GO:0030430; ECO:0000255) matching annotations found in UniProt (1). While it missed the experimentally verified MFO term *identical protein binding* (GO:0042802), goPredSim predicted *protein folding* (GO:0006457) and *protein ubiquitination* (GO:0016567) suggesting the nucleoprotein to be involved in biological processes requiring protein binding. ProNA2020 (3) predicts RNA-binding regions, the one with highest confidence between I84 (Isoleucine at position 84) and D98 (Aspartic Acid at 98) (protein sequence in SOM: Use Case). While high resolution experimental data on binding is not available, an NMR structure of the SARS-CoV-2 nucleocapsid phosphoprotein N-terminal domain in complex with 10mer ssRNA (PDB identifier 7ACT (61)) supports the predicted RNA-binding site (Figure 2). Additionally, SNAP2 (38) predicts single amino acid variants (SAVs) in that region to likely affect function, reinforcing the hypothesis that this is a functionally relevant site. Although using different input information (evolutionary vs. embeddings), RePROF (5) and ProtBertSec (14) both predict an unusual content exceeding 70% non-regular (neither helix nor strand) secondary structure, suggesting that most of the nucleoprotein might not form regular structure. This is supported by Meta-Disorder (30) predicting 53% overall disorder. Secondary structure predictions match well high-resolution experimental structures of the nucleoprotein not in complex (e.g., PDB 6VYO (62); 6WJI (63)). Both secondary structure prediction methods managed to zoom into the ordered regions of the protein and predicted e.g., the five beta-strands that are formed within the sequence range I84 (Isoleucine) to A134 (Alanine), and the two helices formed within the sequence range spanned from F346 (Phenylalanine) to T362 (Tyrosine).

CONCLUSION

For almost three decades (preceding Google) *PredictProtein* (PP) has been offering predictions for proteins. PP is the oldest prediction Internet server, online for 6-times as long as most other servers (64–66). It pioneered combining machine learning with evolutionary information and making predictions freely accessible online. While the sequence-annotation gap continues to grow, the sequence-structure gap might be bridged in the near future (67). For the time being, servers such as PP, providing a one-stop solution to predictions from many sustained, novel tools are needed. Now, PP is the first server to offer fast embedding-based predictions of structure (ProtBertSec) and function (goPredSim). By slashing runtime for PSSMs from 72 to 4 min through MMseqs2 and better LCSB hardware, PP also delivers evolutionary information-based predictions fast! Instantaneously if the query is in the precomputed *PPcache*. For heavy use, the offline Docker containers provide predictors out-of-the-box. At no cost to users, *PredictProtein* offers to quickly shine light on proteins for which little is known using well validated prediction methods.

DATA AVAILABILITY

Freely accessible webserver PredictProtein.org; Source and docker images: github.com/rostlab.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

Maintaining *PredictProtein* over three decades has been tough; many colleagues have helped with hands and brains, developers, and users alike. Thanks to all of you! Please find most contributors in Supplementary Table S2 or at predictprotein.org/credits. In particular, thanks to Noua Toukourou and Maharshi Vyas (both LCSB) for invaluable

help with hardware and software; to David Hoksza (Charles U, Prague) for his work on MolArt; to Marco Punta (IRCCS Milano) for his long-term support; to Inga Weise (TUM) for support with many aspects; to Roy Omond (Blue Bubble, Cambridge), Antoine de Daruvar (Univ. Bordeaux), Yanay Ofran (Bar-Ilan Univ.), Jinfeng Liu (Genentech), Tobias Hamp, Maximilian Hecht, Edda Kloppmann (all previously TUM) for contributing methods and code in the past; Johannes Söding for providing resources to develop and maintain MMseqs2.

FUNDING

Michael Bernhofer was supported by the Competence Network for Scientific High Performance Computing in Bavaria [KONWIHR-III BG.DAF]; Christian Dallago is supported by the Deutsche Forschungsgemeinschaft (DFG) [RO 1320/4-1]; Bundesministerium für Bildung und Forschung (BMBF) [031L0168]; Software Campus 2.0 (TU München), BMBF [01IS17049]; Milot Mirdita acknowledges support from the ERC's Horizon 2020 Framework Programme [`Virus-X', project no. 685778]; BMBF CompLifeSci project horizontal4meta. Martin Steinegger acknowledges support from the National Research Foundation of Korea grant funded by the Korean government (MEST) [2019R1A6A1A10073437, NRF-2020M3A9G7103933]; Creative-Pioneering Researchers Program through Seoul National University; Nir Ben-Tal acknowledges the support of Israeli Science Foundation (ISF) [450/16]; Abraham E. Kazan Chair in Structural Biology, Tel Aviv University; Haim Ashkenazy was supported by Humboldt Research Fellowship for Postdoctoral Researchers of the Alexander von Humboldt Foundation; The PredictProtein web server is hosted by ELIXIR-LU, the Luxembourgish node of the European life-science infrastructure. Funding for open access charge: Library of the Technical University of Munich.
Conflict of interest statement. None declared.

REFERENCES

1. The UniProt Consortium (2021) UniProt: the universal protein knowledgebase in 2021. *Nucleic. Acids Res.*, **49**, D480–D489.
2. 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.
3. Qiu,J., Bernhofer,M., Heinzinger,M., Kemper,S., Norambuena,T., Melo,F. and Rost,B. (2020) ProNA2020 predicts protein–DNA, protein–RNA, and protein–protein binding proteins and residues from sequence. *J. Mol. Biol.*, **432**, 2428–2443.
4. Rost,B. (1996) PHD: predicting one-dimensional protein structure by profile-based neural networks. *Methods Enzymol.*, **266**, 525–539.
5. Yachdav,G., Kloppmann,E., Kajan,L., Hecht,M., Goldberg,T., Hamp,T., Hönigschmid,P., Schafferhans,A., Roos,M., Bernhofer,M. *et al.* (2014) PredictProtein—an open resource for online prediction of protein structural and functional features. *Nucleic Acids Res.*, **42**, W337–W343.
6. Steinegger,M. and Söding,J. (2018) Clustering huge protein sequence sets in linear time. *Nat. Commun.*, **9**, 2542.
7. Rost,B. and Sander,C. (1993) Improved prediction of protein secondary structure by use of sequence profiles and neural networks. *Proc. Natl. Acad. Sci. U.S.A.*, **90**, 7558–7562.
8. Rost,B. and Sander,C. (1993) Prediction of protein secondary structure at better than 70% accuracy. *J. Mol. Biol.*, **232**, 584–599.
9. Rost,B. and Sander,C. (1992) Jury returns on structure prediction. *Nature*, **360**, 540.
10. Kajan,L., Yachdav,G., Vicedo,E., Steinegger,M., Mirdita,M., Angermuller,C., Bohm,A., Domke,S., Ertl,J., Mertes,C. *et al.* (2013) Cloud prediction of protein structure and function with PredictProtein for Debian. *Biomed. Res. Int.*, **2013**, 398968.
11. Steinegger,M. and Söding,J. (2017) MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat. Biotechnol.*, **35**, 1026–1028.
12. Mirdita,M., Steinegger,M. and Söding,J. (2019) MMseqs2 desktop and local web server app for fast, interactive sequence searches. *Bioinformatics*, **35**, 2856–2858.
13. 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 Bioinformatics*, **20**, 723.
14. Elnaggar,A., Heinzinger,M., Dallago,C., Rihawi,G., Wang,Y., Jones,L., Gibbs,T., Feher,T., Angerer,C., Bhowmik,D. *et al.* (2020) ProtTrans: Towards cracking the language of life's code through self-supervised deep learning and high performance computing. arXiv doi: https://arxiv.org/abs/2007.06225, 04 May 2021, preprint: not peer reviewed.
15. Alley,E.C., Khimulya,G., Biswas,S., AlQuraishi,M. and Church,G.M. (2019) Unified rational protein engineering with sequence-based deep representation learning. *Nat. Methods*, **16**, 1315–1322.
16. AlQuraishi,M. (2019) End-to-end differentiable learning of protein structure. *Cell Syst.*, **8**, 292–301.
17. Rao,R., Bhattacharya,N., Thomas,N., Duan,Y., Chen,P., Canny,J., Abbeel,P. and Song,Y. (2019) Evaluating Protein Transfer Learning with TAPE. In: Wallach,H., Larochelle,H., Beygelzimer,A., d'Alché-Buc,F., Fox,E. and Garnett,R. (eds). *Advances in Neural Information Processing Systems*. Vol. **32**. Curran Associates, Inc., pp. 9689–9701.
18. Rives,A., Meier,J., Sercu,T., Goyal,S., Guo,D., Lin,Z., Liu,J., Guo,D., Ott,M., Zitnick,C.L., Ma,J. and Fergus,R. (2021) Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proc. Natl. Acad. Sci. USA*, **118**, e2016239118.
19. 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.
20. Rao,R., Ovchinnikov,S., Meier,J., Rives,A. and Sercu,T. (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.
21. O'Donoghue,S.I., Sabir,K.S., Kalemanov,M., Stolte,C., Wellmann,B., Ho,V., Roos,M., Perdigão,N., Buske,F.A., Heinrich,J. *et al.* (2015) Aquaria: simplifying discovery and insight from protein structures. *Nat. Methods*, **12**, 98–99.
22. Hoksza,D., Gawron,P., Ostaszewski,M. and Schneider,R. (2018) MolArt: a molecular structure annotation and visualization tool. *Bioinformatics*, **34**, 4127–4128.
23. 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.* (2020) SARS-CoV-2 structural coverage map reveals state changes that disrupt host immunity bioinformatics. bioRxiv doi: https://doi.org/10.1101/2020.07.16.207308, 28 September 2020, preprint: not peer reviewed.
24. Altschul,S.F., Gish,W., Miller,W., Myers,E.W. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
25. Altschul,S.F., Madden,T.L., Schäffer,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.
26. Berman,H.M., Westbrook,J., Feng,Z., Gilliland,G., Bhat,T.N., Weissig,H., Shindyalov,I.N. and Bourne,P.E. (2000) The Protein Data Bank. *Nucleic Acids Res.*, **28**, 235–242.
27. Rost,B. (2001) Protein secondary structure prediction continues to rise. *J. Struct. Biol.*, **134**, 204–218.
28. Bernhofer,M., Kloppmann,E., Reeb,J. and Rost,B. (2016) TMSEG: novel prediction of transmembrane helices. *Proteins*, **84**, 1706–1716.
29. Bigelow,H. and Rost,B. (2006) PROFtmb: a web server for predicting bacterial transmembrane beta barrel proteins. *Nucleic Acids. Res.*, **34**, W186–W188.

30. Schlessinger,A., Punta,M., Yachdav,G., Kajan,L. and Rost,B. (2009) Improved disorder prediction by combination of orthogonal approaches. *PLoS One*, **4**, e4433.
31. Schlessinger,A., Yachdav,G. and Rost,B. (2006) PROFbval: predict flexible and rigid residues in proteins. *Bioinforma. Oxf. Engl.*, **22**, 891–893.
32. Ceroni,A., Passerini,A., Vullo,A. and Frasconi,P. (2006) DISULFIND: a disulfide bonding state and cysteine connectivity prediction server. *Nucleic Acids Res.*, **34**, W177–W181.
33. Berezin,C., Glaser,F., Rosenberg,J., Paz,I., Pupko,T., Fariselli,P., Casadio,R. and Ben-Tal,N. (2004) ConSeq: the identification of functionally and structurally important residues in protein sequences. *Bioinforma. Oxf. Engl.*, **20**, 1322–1324.
34. Ashkenazy,H., Erez,E., Martz,E., Pupko,T. and Ben-Tal,N. (2010) ConSurf 2010: calculating evolutionary conservation in sequence and structure of proteins and nucleic acids. *Nucleic Acids Res.*, **38**, W529–W533.
35. Celniker,G., Nimrod,G., Ashkenazy,H., Glaser,F., Martz,E., Mayrose,I., Pupko,T. and Ben-Tal,N. (2013) ConSurf: using evolutionary data to raise testable hypotheses about protein function. *Isr. J. Chem.*, **53**, 199–206.
36. Ashkenazy,H., Abadi,S., Martz,E., Chay,O., Mayrose,I., Pupko,T. and Ben-Tal,N. (2016) ConSurf 2016: an improved methodology to estimate and visualize evolutionary conservation in macromolecules. *Nucleic Acids Res.*, **44**, W344–W350.
37. Sigrist,C.J.A., de Castro,E., Cerutti,L., Cuche,B.A., Hulo,N., Bridge,A., Bougueleret,L. and Xenarios,I. (2013) New and continuing developments at PROSITE. *Nucleic Acids Res.*, **41**, D344–347.
38. Hecht,M., Bromberg,Y. and Rost,B. (2015) Better prediction of functional effects for sequence variants. *BMC Genomics*, **16 (Suppl 8)**, S1.
39. Goldberg,T., Hecht,M., Hamp,T., Karl,T., Yachdav,G., Ahmed,N., Altermann,U., Angerer,P., Ansorge,S., Balasz,K. *et al.* (2014) LocTree3 prediction of localization. *Nucleic Acids Res.*, **42**, W350–W355.
40. El-Gebali,S., Mistry,J., Bateman,A., Eddy,S.R., Luciani,A., Potter,S.C., Qureshi,M., Richardson,L.J., Salazar,G.A., Smart,A. *et al.* (2019) The Pfam protein families database in 2019. *Nucleic Acids Res.*, **47**, D427–D432.
41. Zhou,N., Jiang,Y., Bergquist,T.R., Lee,A.J., Kacsoh,B.Z., Crocker,A.W., Lewis,K.A., Georghiou,G., Nguyen,H.N., Hamid,M.N. *et al.* (2019) The CAFA challenge reports improved protein function prediction and new functional annotations for hundreds of genes through experimental screens. *Genome Biol.*, **20**, 244.
42. Jiang,Y., Oron,T.R., Clark,W.T., Bankapur,A.R., D'Andrea,D., Lepore,R., Funk,C.S., Kahanda,I., Verspoor,K.M., Ben-Hur,A. *et al.* (2016) An expanded evaluation of protein function prediction methods shows an improvement in accuracy. *Genome Biol.*, **17**, 184.
43. Camon,E., Magrane,M., Barrell,D., Lee,V., Dimmer,E., Maslen,J., Binns,D., Harte,N., Lopez,R. and Apweiler,R. (2004) The Gene Ontology Annotation (GOA) Database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Res.*, **32**, D262–D266.
44. Huntley,R.P., Sawford,T., Mutowo-Meullenet,P., Shypitsyna,A., Bonilla,C., Martin,M.J. and O'Donovan,C. (2015) The GOA database: gene ontology annotation updates for 2015. *Nucleic Acids Res.*, **43**, D1057–D1063.
45. El-Mabrouk,N. and Slonim,D.K. (2020) ISMB 2020 proceedings. *Bioinformatics*, **36**, i1–i2.
46. Steinegger,M., Mirdita,M. and Söding,J. (2019) Protein-level assembly increases protein sequence recovery from metagenomic samples manyfold. *Nat. Methods*, **16**, 603–606.
47. Abriata,L.A., Tamò,G.E., Monastyrskyy,B., Kryshtafovych,A. and Peraro,M.D. (2018) Assessment of hard target modeling in CASP12 reveals an emerging role of alignment-based contact prediction methods. *Proteins Struct. Funct. Bioinforma.*, **86**, 97–112.
48. Klausen,M.S., Jespersen,M.C., Nielsen,H., Jensen,K.K., Jurtz,V.I., Sønderby,C.K., Sommer,M.O.A., Winther,O., Nielsen,M., Petersen,B. *et al.* (2019) NetSurfP-2.0: improved prediction of protein structural features by integrated deep learning. *Proteins Struct. Funct. Bioinforma.*, **87**, 520–527.
49. Hamp,T., Goldberg,T. and Rost,B. (2013) Accelerating the original profile kernel. *PLoS One*, **8**, e68459.
50. Asgari,E., McHardy,A.C. and Mofrad,M.R.K. (2019) Probabilistic variable-length segmentation of protein sequences for discriminative motif discovery (DiMotif) and sequence embedding (ProtVecX). *Sci. Rep.*, **9**, 3577.
51. Norambuena,T. and Melo,F. (2010) The protein-DNA interface database. *BMC Bioinformatics*, **11**, 262.
52. Lewis,B.A., Walia,R.R., Terribilini,M., Ferguson,J., Zheng,C., Honavar,V. and Dobbs,D. (2011) PRIDB: a protein-RNA interface database. *Nucleic. Acids. Res.*, **39**, D277–D282.
53. Hamp,T. and Rost,B. (2015) Evolutionary profiles improve protein-protein interaction prediction from sequence. *Bioinforma. Oxf. Engl.*, **31**, 1945–1950.
54. Mirdita,M., von den Driesch,L., Galiez,C., Martin,M.J., Söding,J. and Steinegger,M. (2017) Uniclust databases of clustered and deeply annotated protein sequences and alignments. *Nucleic. Acids. Res.*, **45**, D170–D176.
55. Watkins,X., Garcia,L.J., Pundir,S., Martin,M.J. and Consortium,U. (2017) ProtVista: visualization of protein sequence annotations. *Bioinformatics*, **33**, 2040–2041.
56. Reguant,R., Antipin,Y., Sheridan,R., Dallago,C., Diamantoukos,D., Luna,A., Sander,C. and Gauthier,N.P. (2020) AlignmentViewer: sequence analysis of large protein families. *F1000Research*, **9**, 213.
57. Dallago,C., Goldberg,T., Andrade-Navarro,M.A., Alanis-Lobato,G. and Rost,B. (2020) Visualizing human protein-protein interactions and subcellular localizations on cell images through CellMap. *Curr. Protoc. Bioinforma.*, **69**, e97.
58. Dallago,C., Schütze,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. Bioinforma.*, **1**, e113.
59. Merkel,D. (2014) Docker: lightweight linux containers for consistent development and deployment. *Linux J.*, **2014**, 2.
60. MacDougall,A., Volynkin,V., Saidi,R., Poggioli,D., Zellner,H., Hatton-Ellis,E., Joshi,V., O'Donovan,C., Orchard,S., Auchincloss,A.H. *et al.* (2020) UniRule: a unified rule resource for automatic annotation in the UniProt Knowledgebase. *Bioinformatics*, **36**, 4643–4648.
61. Dinesh,D.C., Chalupska,D., Silhan,J., Koutna,E., Nencka,R., Veverka,V. and Boura,E. (2020) Structural basis of RNA recognition by the SARS-CoV-2 nucleocapsid phosphoprotein. *PLoS Pathog.*, **16**, e1009100.
62. Chang,C., Michalska,K., Jedrzejczak,R., Maltseva,N., Endres,M., Godzik,A., Kim,Y. and Joachimiak,A. (2020) Crystal structure of RNA binding domain of nucleocapsid phosphoprotein from SARS coronavirus 2. doi:10.2210/pdb6vyo/pdb.
63. Minasov,G., Shuvalova,L., Wiersum,G. and Satchell,K. (2020) 2.05 angstrom resolution crystal structure of C-terminal dimerization domain of nucleocapsid phosphoprotein from SARS-CoV-2. doi:10.2210/pdb6wji/pdb.
64. Schultheiss,S.J., Münch,M.-C., Andreeva,G.D. and Rätsch,G. (2011) Persistence and availability of Web services in computational biology. *PLoS One*, **6**, e24914.
65. Wren,J.D., Georgescu,C., Giles,C.B. and Hennessey,J. (2017) Use it or lose it: citations predict the continued online availability of published bioinformatics resources. *Nucleic Acids. Res.*, **45**, 3627–3633.
66. Kern,F., Fehlmann,T. and Keller,A. (2020) On the lifetime of bioinformatics web services. *Nucleic Acids Res.*, **48**, 12523–12533.
67. Callaway,E. (2020) 'It will change everything': DeepMind's AI makes gigantic leap in solving protein structures. *Nature*, **588**, 203–204.

5.4 FLIP: Benchmark tasks in fitness landscape inference for proteins

Summary. While representation learning approaches like pLMs could unlock protein design applications, no benchmark assessing their native ability to do so existed. Developing sets probing the ability of machine learning to design proteins is challenging, as some proteins are multi-purpose molecules, and current biological experiments often focus on single aspects of selected proteins. While systematic approaches like CASP and CAFA, assessing structure and function systematically respectively exist, they do not target metrics relevant for protein engineering. Fitness Landscape Inference for Proteins (FLIP) is a curated set of several biological experiments aimed at probing the ability of machine representations of proteins to support protein design campaigns. To achieve this, several *splits* from three experimental datasets were devised, testing the ability of protein representations to emulate typical experimental protein design settings, e.g., extrapolative (predicting the effect of multiple changes along the protein sequence by knowing the effect of few changes) and low-resource (predicting landscapes from only a few labelled samples). The landscape splits come with data standardization, enabling quick adoption in computational pipelines, and enabling easy probing for new representation models.

Relevance. While probing pLMs' ability to characterize protein sequences by predicting on traditionally accepted tasks such as structure and well-defined aspect of function (e.g., subcellular localization (Stärk *et al.*, 2021)) may support their validity, these annotations are *sharp* cutouts of the "continuous" nature of protein function that may need to be captured to design proteins. Embeddings from pLMs encode continuous representations that could potentially correlate with the continuous nature of function. One attempt to correlate these realities is to predict mutational landscapes using embeddings (Marquet *et al.*, 2021). However, probing purely on deep mutational scanning (DMS) sets limited to mutational effects of single residue substitutions one at the time may not entirely characterize more complex mutational neighborhoods from a wildtype sequence. Experiments introducing a random number of residue substitutions offer a complementary approach to DMS sets. FLIP contributes by introducing four datasets for the assessment of protein representations to stack up to the continuous nature of protein function. Two of the three datasets focused on mutational landscapes from a wildtype sequence to mutated versions of it with up to 32 changes. The last dataset focused on protein thermal stability, characterizing the turning degree at which proteins start to denature (i.e., become ineffective).

Contribution. I am one of two principal authors of this paper. I contributed conceptualization, implementation, and writing.

Copyright notice. The original publication is available in open access at openreview.net/forum?id=p2dMLEwL8tF and in the following. The copyright notice is attached in this appendix after the manuscript.

FLIP: Benchmark tasks in fitness landscape inference for proteins

Christian Dallago[*]
Technical University of Munich
christian.dallago@tum.de

Jody Mou[*]
Microsoft Research New England
jodymou@mit.edu

Kadina E. Johnston
BBE, Caltech
kjohnston@caltech.edu

Bruce J. Wittmann
BBE, Caltech
bwittman@caltech.edu

Nicholas Bhattacharya
UC Berkeley
nick_bhat@berkeley.edu

Samuel Goldman
CSB, MIT
samlg@mit.edu

Ali Madani
Saleforce Research
amadani@salesforce.com

Kevin K. Yang
Microsoft Research New England
yang.kevin@microsoft.com

Abstract

Machine learning could enable an unprecedented level of control in protein engineering for therapeutic and industrial applications. Critical to its use in designing proteins with desired properties, machine learning models must capture the protein sequence-function relationship, often termed *fitness landscape*. Existing benchmarks like CASP or CAFA assess structure and function predictions of proteins, respectively, yet they do not target metrics relevant for protein engineering. In this work, we introduce Fitness Landscape Inference for Proteins (FLIP), a benchmark for function prediction to encourage rapid scoring of representation learning for protein engineering. Our curated tasks, baselines, and metrics probe model generalization in settings relevant for protein engineering, e.g. low-resource and extrapolative. Currently, FLIP encompasses experimental data across adeno-associated virus stability for gene therapy, protein domain B1 stability and immunoglobulin binding, and thermostability from multiple protein families. In order to enable ease of use and future expansion to new tasks, all data are presented in a standard format. FLIP scripts and data are freely accessible at https://benchmark.protein.properties.

1 Introduction

Proteins are life's workhorses, efficiently and precisely performing complex tasks under a wide variety of conditions. This combination of versatility and selectivity makes them not only critical to life, but also to a myriad of human-designed applications. Engineered proteins play increasingly essential roles in industries and applications spanning pharmaceuticals, agriculture, specialty chemicals, and fuel [1–5]. The ability of a protein to perform a desired function is determined by its amino acid sequence, often mediated through folding to a three-dimensional structure [6]. Unfortunately, current biophysical and structural prediction methods cannot reliably map a sequence to its ability to perform a desired function, termed protein *fitness*, with sufficient precision to distinguish between closely-related protein sequences performing complex functions such as catalysis. Therefore, protein engineering has relied heavily on directed evolution (DE) methods, which stochastically modify ("mutate") a starting sequence to create a library of sequence variants, measure all variants to find those with improved fitness, and then iterate until the protein is sufficiently optimized [7]. Directed evolution is energy-, time-, and material-intensive, in part because it discards information from

[*] Equal contribution

unimproved sequences. Machine-learning methods that predict fitness from sequence can leverage both positive and negative data to intelligently select variants for screening, reaching higher fitness levels with fewer measurements than traditional directed evolution, and without necessarily requiring detailed understanding of structure or mechanism [8, 7, 9–11].

Directed evolution campaigns are often limited by the cost of collecting sequence-fitness data. Therefore, machine learning approaches for sequence-fitness prediction are most useful in protein engineering when they can learn from low-N (few sample) labeled datasets or when they can generalize to types of variation that are unobserved in the training set. Rapid advances in genomic sequencing technology have led to an explosion of putative protein sequences [12, 13] deposited in databases like UniProt [14]. Recent efforts in sequence-function prediction [15, 16] have sought to leverage the information in these unlabeled sequences through pretraining and fine-tuning, and have successfully engineered proteins with brighter fluorescence and high catalytic efficiency [17]. Unsupervised models were also applied to- or built on evolutionary sequence inputs to model the effects of mutations [18–21].

In this work, we present a suite of benchmarking tasks for protein sequence-fitness prediction with the dual aims of enabling protein engineers to compare and choose machine learning methods representing protein sequences and accelerating research on machine learning for protein fitness prediction. Our tasks are curated to be diverse in the functions measured and in the types of underlying sequence variation. For each landscape, we provide one or more train/test splits that evaluate biologically-relevant generalization and mimic challenges often seen in protein engineering. Figure 1 and Table 2 summarize the landscape tasks and splits. We also compute the performance of baseline models against which future models can be compared, and which highlight that our tasks can distinguish between “better" and “worse" pretraining regimes. Landscapes and baselines are available at https://benchmark.protein.properties, while a glossary technical terms is provided in the supplement.

2 Related Work

Well-designed and easily accessible benchmarks have encouraged and measured progress in machine learning on proteins, especially protein structure prediction. The Critical Assessment of Protein Structure Prediction (CASP) [22], and retrospective protein training datasets from previous CASP competitions [23] have lowered the barrier to entry for new research teams and provided a clear account of progress over the last three decades [24]. DeepMind’s recent landmark results with their *AlphaFold2* predictor in CASP 14 [25] built on these community-driven efforts.

Table 1: Performance (Spearman’s correlation) on TAPE engineering tasks. Performances reported in referenced literature. CNNs were replicated from [26] without test set clipping.

	Pretraining	Fluorescence	Stability
ESM [27]	masked language model	0.68	0.71
TAPE transformer [28]	masked language model	0.68	0.73
TAPE LSTM [28]	bidirectional language model	0.67	0.69
TAPE ResNet [28]	masked language model	0.21	0.73
UniRep [29]	language model + structure	0.67	0.73
CPCProt [30]	contrastive	0.68	0.65
CPCProt-LSTM [30]	contrastive	0.68	0.68
Linear regression [26]	none	0.68	0.48
CNN [26]	none	0.67	0.51
Mutation count [31]	none	0.45	NA
BLOSUM62 score [31]	none	0.50	NA

Inspired by the effectiveness of CASP, there have been attempts at benchmarks for function prediction and protein pretraining. The Critical Assessment of Function Annotation (CAFA) [32, 33] focuses on assigning Gene Ontology (GO) classes (categorical definitions of protein functions) to proteins. While an important benchmark, CAFA does not directly require models to build on sequence inputs, instead

they could leverage graph inputs from protein-protein interaction networks, and the prediction targets do not account for fitness variations between very similar sequences that are important for protein engineering. Tasks Assessing Protein Embeddings (TAPE) [28] aims to evaluate the effectiveness of different pretraining regimes and models to predict protein properties. Of the five tasks in TAPE, three (remote homology, secondary structure, and contacts) focus on structure prediction, while only two (fluorescence and stability) target fitness prediction. These two tasks show little discriminative power between different models [26], as shown in Table 1. In addition, the use of structure as an evaluation limits the creation of jointly trained structure- and sequence- based embeddings that may be most useful in protein engineering tasks [34]. Envision [35] collates several dozen single amino-acid variation (SAV) datasets, but does not include other types of sequence variation of interest to protein engineers. DeepSequence [19] collects 42 deep mutational scan (DMS) datasets for evaluation purposes. These capture single and multiple co-occurring residue substitutions, but do not capture variation at the proteome scale, or mutational paths from large insertions and deletions. Furthermore, while DMS landscapes may characterize the effect of co-occurring substitutions, not every sample with co-occurring residue substitutions may express these at sites relevant for a measured function, and in turn, evaluations on all possible co-occurring substitutions may not always be expressive (e.g., if the measured function is binding and a sample has two substitutions, one at a residue at the interface and one elsewhere, the effect may still be high simply because an interface residue is involved). Finally, the data from these studies does not come with standard column headers or train/test splits, hindering use in automated evaluation pipelines.

The limitations of the existing benchmarks have led pretraining methods to be primarily evaluated by their ability to predict structural information [36, 37]. While the ability to impart structural knowledge through sequence-only pretraining is impressive, it is not the most important criterion for protein engineers. Efforts to systematically compare new methods on fitness prediction have required researchers to both gather their own collection of datasets and compute their own baseline comparisons [16, 38–40].

3 Landscapes and Splits

We design FLIP to answer two fundamental questions about machine learning model learning protein sequences:

1. Can a model capture complex fitness landscapes beyond mutations to a parent sequence?
2. Can a model perform well across a range of proteins where fitness is measured for very different functions?

Existing work such as DeepSequence [19] and Envision [35] succeed at the second criterion but not the first. TAPE [28], on the other hand, evaluates the first criterion with its fluorescence task but not the second. We prioritized complex landscapes (with insertions and deletions) rather than single amino acid variants (e.g. deep mutational scans), to practically cover a larger sequence space, as well as potentially more functional diversity finalized to ensure model generalization for broad applicability.

To test the aforementioned questions, we collect three published landscapes and create 15 corresponding dataset splits as desribed in the following and summarized in Table 2. We choose landscapes and splits that cover a broad range of protein families, sequence variation, and fitness landscapes with rigorous measurements. Each landscape is transformed into one or more splits to test different model generalization abilities, as shown in Figure 1; many of the splits were also made to reflect standard laboratory data-collection practices, thus testing the appropriateness of models to real-world applications.

Simple random splits are notoriously misleading in classical protein sequence-to-function prediction as protein sequences are not sampled I.I.D., but with correlations induced by evolutionary history. This means that random splits reflect a notion of generalization not of interest to most biologists [46]. While there are standard heuristics for approximating the correlation structure due to evolution (such as sequence-identity deduplication\redundancy reduction), in the protein engineering setting there are not similarly standardized approaches. As such, we resorted to landscape-specific approaches informed by the conditions of each experiment, as detailed in Figure 1.

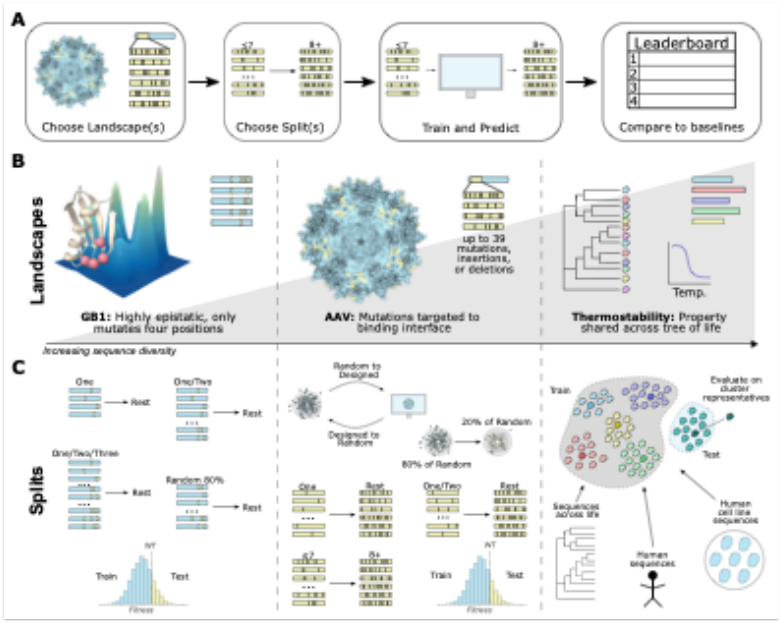


Figure 1: Summary of the workflow, landscapes, and splits. (A) General FLIP workflow: choose landscapes and splits that match user needs, train models and make predictions on the test set, and then compare to baseline models. (B) We choose landscapes that cover different types of sequence diversity. The GB1 landscape focuses on simultaneous mutation of four epistatic sites with nearly complete coverage [41] (PDB ID: 2GI9 [42]). The AAV capsid protein landscape sparsely samples sequences with up to 28 mutations, including insertions and deletions, to the the binding interface [43] (PDB ID: 6IH9 [44]). The thermostability landscape [45] measures a property shared by proteins from multiple functional groups across different domains of life. (C). We also provide up to seven suggested data splits for each landscape, which are described in Section 3.

Table 2: Landscapes and split statistics. The sampled splits (*) are mainly used for discourse in this manuscript, as such splits are rarely observed in practice when working with biological data.

Landscape	Split	Total samples	Train samples	Test samples
AAV	Mut-Des	284,009	82,583	201,426
	Des-Mut	284,009	201,426	82,583
	1-vs-rest	82,583	1,170	81,413
	2-vs-rest	82,583	31,807	50,776
	7-vs-rest	82,583	70,002	12,581
	low-vs-high	82,583	47,546	35,037
	Sampled*	82,583	66,066	16,517
Thermostability	Mixed	27,951	24,817	3,134
	Human	10,093	8,148	1,945
	Human-cell	7,156	5,792	1,366
GB1	1-vs-rest	8,733	29	8,704
	2-vs-rest	8,733	427	8,306
	3-vs-rest	8,733	2,968	5,765
	low-vs-high	8,733	5,089	3,644
	Sampled*	8,733	6,961	1,772

The vast majority of representation learning on protein sequences models entire sequences [27, 37, 15, 34]. As such, we use entire protein sequences as inputs, even for landscapes derived from studies examining mutations at a small subset of positions. While we include a naïve validation set for each split for comparison purposes, we encourage users to engineer their own validation splits from the training data. All tasks and splits are provided in a consistent, easy-to-use CSV format and are available at https://benchmark.protein.properties. Original datasets were either supplemented to published research (Wu et al.) under CC BY 4.0, or were obtained with written permission from the authors (Jarzab et al., Bryant et al.). Data derivatives proposed as tasks are licensed under AFL-3.

3.1 GB1

Motivation. One challenge confronting protein engineering is the ability to predict the effects of interactions between mutations, termed epistasis. These interactions result in non-additive effects on protein fitness and have been shown to constrain the paths available to evolution, especially evolution via a greedy walk. Furthermore, as more mutations are made simultaneously, these interactions become more complex and more difficult to predict. Therefore, we wish to assess model predictions on an exhaustive, combinatorial, and highly epistatic mutational landscape, focusing on learning from variants with fewer mutations to predict the activity of variants with more mutations.

Landscape. We use the GB1 landscape [41], which has become a gold standard for investigating epistatic interactions [10]. GB1 is the binding domain of protein G, an immunoglobulin binding protein found in Streptococcal bacteria [47, 48]. In their original study, Wu et al. measured the fitness of $149,361$ of $160,000$ possible combinations of mutations at 4 positions.

Splits. Over 96% of the amino acid mutations in this set yield non- or poorly-binding sequences – $143,539$ out of $149,361$ sequences have fitness value below 0.5, where wild-type fitness is 1 and a fitness of 0 is non-binding. Thus, models trained on the full experimental data can achieve high performance by predicting low fitness regardless of inputs. To ensure that models learn nontrivial signal, we downsample non-functional sequences prior to creating the training sets. Specifically, we include all 5822 sequences with fitness above 0.5 and 2911 randomly-sampled sequences with fitness less than or equal to 0.5. From this set, we curate five dataset splits to test generalization from few-mutation sequences to many-mutation sequences, from low fitness to high, and one extra randomly sampled split for discussion purposes:

- **Train on single mutants (1-vs-rest):** Wild type and single mutants are assigned to train, while the rest are assigned to test. This split is one of the most commonly observed in an applications setting, where a researcher has gathered data for many single mutations of interest and wishes to predict the best combinations of mutations.
- **Train on single and double mutants (2-vs-rest):** Wild type, single and double mutants are assigned to train, while the rest are assigned to test. This is also a commonly observed split in an applications setting, albeit, at a lesser frequency than 1-vs-rest.
- **Train on single, double and triple mutants (3-vs-rest):** Wild type, single, double and triple mutants are assigned to train, while the rest are assigned to test.
- **Train on low fitness, test on high (low-vs-high):** Sequences with fitness value equal or below wild type are used to train, while sequences with fitness value above wild type are used to test.
- **Sampled:** Sequences are randomly partitioned in 80% train and 20% test. This split serves mostly for discussion purposes in this manuscript.

3.2 AAV

Motivation. Mutations for engineering are often focused in a specific region of a protein. For example, this is done if a protein-protein interface is known to be at a subset of positions. Successfully predicting fitness for a long sequence being mutated at a subset of positions is a task of wide applicability.

Landscape. Adeno-associated virus (AAV) capsid proteins are responsible for helping the virus integrate a DNA payload into a target cell [49], and there is great interest in engineering versions of these proteins for gene therapy [43, 50, 51]. Bryant et al. prepared a rich mutational screening landscape of different VP-1 AAV proteins (UniProt [14] Accession: P03135), and this data has been successfully used as a basis for machine learning-guided design [52, 53]. In their study, Bryant et al. mutagenize a 28-amino acid window from position 561 to 588 of VP-1 and measure the fitness of resulting variants with between 1 and 39 mutations, which we refer to as the *sampled* pool. In addition they measured the fitness of sequences chosen or designed using various machine-learning models. We refer to these as the *designed* pool.

Splits. We derive seven splits from this landscape that probe model generalization:

- **Sampled-designed (Mut-Des):** All *sampled* sequences are assigned to train; all *designed* sequences are assigned to test.
- **Designed-sampled (Des-Mut):** All *designed* sequences are assigned to train; all *sampled* sequences are assigned to test.
- **Train on single mutants (1-vs-rest):** Wild type and single mutants in the *sampled* pool are assigned to train, while the rest are assigned to test. As with the GB1 1-vs-rest split, this reflects a common dataset split observed in protein engineering applications.
- **Train on single and double mutants (2-vs-rest):** Wild type, single and double mutants in the *sampled* pool are assigned to train, while the rest are assigned to test. Again, as with the GB1 2-vs-rest split, this reflects a common dataset split observed in protein engineering applications.
- **Train on mutants with up to seven changes (7-vs-rest):** Mutants with up to and including seven changes in the *sampled* pool are assigned to train, while the rest are assigned to test.
- **Train on low fitness, test on high (low-vs-high):** For sequences in the in the *sampled* pool, sequences with fitness value equal or below wild type are used to train, while sequences with fitness value above wild type are used to test.
- **Sampled:** Sequences in the *sampled* pool are randomly partitioned in 80% train and 20% test. This split serves mostly for discussion purposes in this manuscript.

3.3 Thermostability

Motivation. Thermostability is very often a desirable trait that complements more application-specific functions. For example, thermostable enzymes not only allow operation at higher reaction temperatures with faster reaction rates, but are also better starting points for directed evolution campaigns [54, 55]. This explains why thermostability has been a consistent target for multi-objective optimization in protein engineering [56–58]. Thermostability can be challenging to predict, because it is not necessarily a smooth function landscape; in certain protein families, a single amino acid substitution can confer or destroy thermostability [59].

Landscape. We curate an extensive screening landscape from the Meltome Atlas [45], which used a mass spectrometry-based assay to measure protein melting curves across 13 species and 48,000 proteins. Unlike the other landscapes, which measure the effects of sequence variation from a single starting point on a function specific to that protein, this landscape includes both global and local variation.

Splits. We derive three splits from this landscape, considering biological realities and common dataset regularizations for cross-spices and sequence-diverse sets:

- **Mixed:** We cluster all available sequences and select cluster representatives using MMseqs2 [12] at a threshold of 20% sequence identity to create one split. In this split, all sequences in 80% of clusters are assigned to train, while only cluster representatives from the remaining 20% of clusters are assigned to test.
- **Human:** We cluster sequences in human and select cluster representatives using MMseqs2 [12] at a threshold of 20% sequence identity to create one split. In this split, all

sequences in 80% of clusters are assigned to train, while only cluster representatives from the remaining 20% of clusters are assigned to test.

- **Human-cell:** We cluster sequences of one cell line for human and select cluster representatives using MMseqs2 [12] at a threshold of 20% sequence identity to create one split. In this split, all sequences in 80% of clusters are assigned to train, while only cluster representatives from the remaining 20% of clusters are assigned to test.

4 Baseline algorithms

We evaluate three major groups of baselines (Table 3) – parameter-free, supervised, and pretrained. These three classes correspond to common approaches from different communities. In particular, we seek to clarify the value of transfer learning for protein engineering by benchmarking pretrained models against purely supervised methods systematically. We also hope to simplify algorithm selection for practitioners by providing a single place to compare many commonly used methods. Note that we do not use Potts models [60], popular in protein structure prediction [61], because of the need to build high-quality multiple sequence alignments, which would be impractical for the thermostability dataset. Furthermore, Potts models use artificial constructs when dealing with datasets with large insertions and deletions (e.g., modeling sequence deletions through special characters), as is the case for the AAV landscape. However, in the presence of well curated MSAs, these approaches can be successful in modeling the effect of residue substitutions [62].

Table 3: Baseline methods

Method	Description
Levenshtein	Levenshtein distance to wild-type.
BLOSUM62	BLOSUM62-score relative to wild-type.
Ridge regression	Ridge regression model on one-hot encoding.
Convolutional network	Simple convolutional network on one-hot encoding.
ESM-untrained	750M parameter transformer with randomly-initialized weights
ESM-1b [27]	750M parameter transformer pretrained on UniRef50.
ESM-1v [16]	750M parameter transformer pretrained on UniRef90. Only **one** element of ensemble used due to compute constraints.

For baselines using protein language models, which compute an embedding for every amino acid, we pool embeddings in three ways:

- **Per amino acid (per AA):** A supervised model is tasked to learn how to pool over the sequence using a 1D attention layer to return a regression prediction.
- **Mean:** Sequence embeddings are mean pooled per amino acid over the length of the protein sequence to obtain a fixed-size input for each sequence.
- **Mean over subset (mut mean):** Sequence embeddings are mean pooled per amino acid for the residues in the mutated region of interest to obtain a fixed-size, region specific input from the sequence.

To train the models, 10% of each training set is sampled at random as a validation set. For *Ridge*, we use the scikit-learn implementation of ridge regression with default parameters. The CNN consists of a convolution with kernel width 5 and 1024 channels, a ReLU non-linearity, a linear mapping to 2048 dimensions, max pool over the sequence, and a linear mapping to 1 dimension. CNNs are optimized using Adam [63] with a batch size of 256 (GB1, AAV) or 32 (thermostability) and a learning rate of 0.001 for the convolution weights, 0.00005 for the first linear mapping, and 0.000005 for the second linear mapping. Both linear mappings have a weight decay of 0.05. For ESM models, by far the most computationally expensive baselines, we train with a batch size of 256, a learning rate of 0.001, and the Adam optimizer. CNNs and the ESM models are trained with early stopping with a patience of 20 epochs. Models are trained on a NVidia Quadro RTXA6000 GPU. Code, data, and instructions needed to reproduce results can be found at https://benchmark.protein.properties.

5 Results

Overall, we observe that for landscapes around a wild type (Tables 4 & 5), pretraining offered by ESM-1b [27] or ESM-1v [16] does not help much when sufficient training data is available (see Table 2 for statistics), at least in the setting explored here: using these protein language models to collect *frozen* embeddings as inputs to subsequent prediction models. Conversely, for the split involving diverse sequences (Table 6), pretraining yields a large boost over pure supervision. The best method of pooling residue-embeddings for whole sequences varies depending on task (Table 4, 5 & 6). Most remarkably, training simple models (CNN, ridge regression) is competitive over a wide range of regimes. We exclude results for *per-AA* ESM models for the AAV *Des-Mut* task (Table 5), as we estimated that it would require a month of compute using for Nvidia A6000 GPUs, which appeared unjustified for a baseline metric computation. Hyperparameter search results are reported in the supplement, as are evaluations using different metrics.

Table 4: GB1 baselines (metric: Spearman correlation)

Model	**1-vs-rest**	**2-vs-rest**	**3-vs-rest**	**low-vs-high**
ESM-1b (per AA)	0.28	0.55	0.79	0.59
ESM-1b (mean)	0.32	0.36	0.54	0.13
ESM-1b (mut mean)	-0.08	0.19	0.49	0.45
ESM-1v (per AA)	0.28	0.28	0.82	0.51
ESM-1v (mean)	0.32	0.32	0.77	0.10
ESM-1v (mut mean)	0.19	0.19	0.80	0.49
ESM-untrained (per AA)	0.06	0.06	0.48	0.23
ESM-untrained (mean)	0.05	0.05	0.46	0.10
ESM-untrained (mut mean)	0.21	0.21	0.57	0.13
Ridge	0.28	0.59	0.76	0.34
CNN	0.17	0.32	0.83	0.51
Levenshtein	0.17	0.16	-0.04	-0.10
BLOSUM62	0.15	0.14	0.01	-0.13

GB1. Table 4 summarizes baseline results for the biologically motivated GB1 splits. When models are trained only on single mutants, all variations on ESM-1b [27] and ESM-1v [16] outperform supervised models. This regime has little training data (29 samples, Table 2), giving the most opportunity for pretraining to compensate. The difference between pretrained and supervised models largely disappears once models are trained on both single and double mutants (2-vs-rest, Table 4). The various pooling choices for embeddings perform inconsistently across datasets and splits; for example, *mut-mean* does best on 1-vs-rest but worst on 3-vs-rest. The *low-vs-high* split suggests. The sampled split reported separately in Table 7 confirms: random sampling sequences in biology is bound to overestimate results.

AAV. Table 5 summarizes baseline results for the biologically motivated AAV splits. Across all splits, purely supervised models are competitive with pretrained models. This suggests that the large sizes of training sets are past the threshold where pretraining improves performance. The particular choice of pooling that performs best is inconsistent across splits. The BLOSUM62 baseline could not be applied as the mutations in this set include insertions and deletions. In this case too, the sampled split reported separately in Table 7 strongly suggest that random sampling sequences in biology may lead to overestimated results.

Thermostability. Table 6 summarizes baseline results for thermostability. Pretrained models consistently outperform supervised models on this task, suggesting that this landscape is not yet past the threshhold where pretraining improves performance. Interestingly, the supervised baselines based on untrained ESM embeddings do better than either ridge or CNN. Mean over subset (mut mean) and BLOSUM62 are not applicable for the Meltome landscape as the sequences are not evolutionarily related.

Table 5: AAV baselines (metric: Spearman correlation)

Model	Mut-Des	Des-Mut	1-vs-rest	2-vs-rest	7-vs-rest	low-vs-high
ESM-1b (per AA)	0.76	—	0.03	0.65	0.65	0.39
ESM-1b (mean)	0.63	0.59	0.04	0.26	0.46	0.18
ESM-1b (mut mean)	0.70	0.70	0.31	0.65	0.61	0.33
ESM-1v (per AA)	0.79	—	0.10	0.70	0.70	0.34
ESM-1v (mean)	0.55	0.44	0.18	0.16	0.45	0.20
ESM-1v (mut mean)	0.70	0.71	0.44	0.64	0.64	0.31
ESM-untrained (per AA)	0.56	—	0.18	0.22	0.42	0.08
ESM-untrained (mean)	0.27	0.34	0.01	0.14	0.22	0.22
ESM-untrained (mut mean)	0.62	0.64	0.26	0.16	0.56	0.24
Ridge	0.64	0.53	0.22	0.03	0.65	0.12
CNN	0.71	0.75	0.48	0.74	0.74	0.34
Levenshtein	0.60	-0.07	-0.11	0.57	0.53	0.25
BLOSUM62	NA	NA	NA	NA	NA	NA

Table 6: Thermostability baselines (metric: Spearman correlation)

Model	Mixed	Human	Human-Cell
ESM-1b (per AA)	0.68	0.71	0.76
ESM-1b (mean)	0.68	0.70	0.75
ESM-1b (mut mean)	NA	NA	NA
ESM-1v (per AA)	0.65	0.77	0.78
ESM-1v (mean)	0.67	0.75	0.74
ESM-1v (mut mean)	NA	NA	NA
ESM-untrained (per AA)	0.44	0.44	0.46
ESM-untrained (mean)	0.36	0.48	0.49
ESM-untrained (mut mean)	NA	NA	NA
Ridge	0.17	0.15	0.24
CNN	0.34	0.50	0.49
Levenshtein	NA	NA	NA
BLOSUM62	NA	NA	NA

6 Discussion

The prediction tasks in FLIP probe complex fitness landscapes across different protein functions. We curate three landscapes published in existing literature and formulate 15 corresponding splits of the data to mimic protein engineering tasks. The main criteria to include a landscape was whether it could be used to assess interesting types of generalization, and if it was amenable to interpretable assessment metrics. As no standard approach exists to partition landscapes arising from mutagenesis of a parent sequence, we propose ideas that may be applied to future landscapes. In particular, we explore the concept of training on sequences only a few mutations from a parent while predicting on data many mutations from a parent in a step-by-step fashion.

The need for more challenging splits is illustrated in Table 7, which shows results for the *sampled* splits, based on simple random sampling. Almost all models do drastically better for the sampled splits, and differences between models are exaggerated. This indicates the importance of biologically-motivated generalization in task design.

In general, results on baselines highlight that while pretraining approaches perform well on tasks with diverse sequences (Thermostability, Table 6), they do not outperform simpler models on mutational landscapes (GB1, Table 4 &, AAV, Table 5). In addition, large pretrained models require amounts of compute (up to 50 days on an NVidia A6000 GPU) to train on some tasks, which is out of the reach of most academic research groups. It is important to note that while we performed a modest hyperparameter search, more extensive sweeps combined with training data regularization

Table 7: Optimistic results for random splits (*Sampled)* on the AAV and GB1 sets (metric: Spearman correlation)

Landscape	**AAV**	**GB1**
ESM-1b (per AA)	0.90	0.92
ESM-1v (per AA)	0.92	0.92
ESM-untrained (per AA)	0.78	0.79
Ridge	0.83	0.82
CNN	0.92	0.91

like different validation splits, may yield better absolute and relative performance. The landscapes and derived prediction splits offered in FLIP highlight directions for future work, such as better pretraining or embedding methods for protein mutational landscapes.

7 Conclusion

The proliferation of protein sequence data, along with advanced experimental techniques for functional measurement of proteins, presents a ripe environment for machine learning-enabled solutions in protein engineering. With the introduction of FLIP, we focus on sequence-fitness prediction and aim to encourage rigorous evaluation of model generalization in multiple tasks and settings relevant to protein engineering. We hope to seed advances in this emerging interdisciplinary field with downstream applications for solutions in human health and the environment. FLIP data and scripts are available under free licenses at https://benchmark.protein.properties.

Acknowledgments and Disclosure of Funding

The authors thank Jeffrey Spencer, Sam Sinai, Sam Bowman, Roshan Rao and Debora Marks for ideas and discussions that helped us improve our work. The authors would also like to thank Helix and Murphy for careful attention to the manuscript. C.D. acknowledges support from the Bundesministerium für Bildung und Forschung (BMBF) – Project numbers: 01IS17049 and 031L0168. K.E.J. and B.J.W. acknowledge the NSF Division of Chemical, Bioengineering, Environmental and Transport Systems (1937902). N.B. was supported in part by NIH grant R35-GM134922 and by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. S.G. thanks the MIT Machine Learning for Pharmaceutical Discovery and Synthesis Consortium for supporting this work. K.K.Y. was previously employed by Generate Biomedicines.

References

[1] Stephan Luetz, Lori Giver, and James Lalonde. Engineered enzymes for chemical production. *Biotechnology and Bioengineering*, 101(4):647–653, August 2008. doi: 10.1002/bit.22077. URL `https://doi.org/10.1002/bit.22077`. 1

[2] Fei Wen, Nikhil U Nair, and Huimin Zhao. Protein engineering in designing tailored enzymes and microorganisms for biofuels production. *Current Opinion in Biotechnology*, 20(4):412–419, August 2009. doi: 10.1016/j.copbio.2009.07.001. URL `https://doi.org/10.1016/j.copbio.2009.07.001`.

[3] Swati Kapoor, Aasima Rafiq, and Savita Sharma. Protein engineering and its applications in food industry. *Critical Reviews in Food Science and Nutrition*, 57(11):2321–2329, June 2015. doi: 10.1080/10408398.2014.1000481. URL `https://doi.org/10.1080/10408398.2014.1000481`.

[4] Mark A. Huffman, Anna Fryszkowska, Oscar Alvizo, Margie Borra-Garske, Kevin R. Campos, Keith A. Canada, Paul N. Devine, Da Duan, Jacob H. Forstater, Shane T. Grosser, Holst M. Halsey, Gregory J. Hughes, Junyong Jo, Leo A. Joyce, Joshua N. Kolev, Jack Liang, Kevin M. Maloney, Benjamin F. Mann, Nicholas M. Marshall, Mark McLaughlin, Jeffrey C. Moore,

Grant S. Murphy, Christopher C. Nawrat, Jovana Nazor, Scott Novick, Niki R. Patel, Agustina Rodriguez-Granillo, Sandra A. Robaire, Edward C. Sherer, Matthew D. Truppo, Aaron M. Whittaker, Deeptak Verma, Li Xiao, Yingju Xu, and Hao Yang. Design of an in vitro biocatalytic cascade for the manufacture of islatravir. *Science*, 366(6470):1255–1259, 2019. ISSN 0036-8075. doi: 10.1126/science.aay8484. URL https://science.sciencemag.org/content/366/6470/1255.

[5] Carlos Eduardo Sequeiros-Borja, Bartłomiej Surpeta, and Jan Brezovsky. Recent advances in user-friendly computational tools to engineer protein function. *Briefings in Bioinformatics*, 22 (3), July 2020. doi: 10.1093/bib/bbaa150. URL https://doi.org/10.1093/bib/bbaa150. 1

[6] Predrag Radivojac, Wyatt T Clark, Tal Ronnen Oron, Alexandra M Schnoes, Tobias Wittkop, Artem Sokolov, Kiley Graim, Christopher Funk, Karin Verspoor, Asa Ben-Hur, Gaurav Pandey, Jeffrey M Yunes, Ameet S Talwalkar, Susanna Repo, Michael L Souza, Damiano Piovesan, Rita Casadio, Zheng Wang, Jianlin Cheng, Hai Fang, Julian Gough, Patrik Koskinen, Petri Törönen, Jussi Nokso-Koivisto, Liisa Holm, Domenico Cozzetto, Daniel W A Buchan, Kevin Bryson, David T Jones, Bhakti Limaye, Harshal Inamdar, Avik Datta, Sunitha K Manjari, Rajendra Joshi, Meghana Chitale, Daisuke Kihara, Andreas M Lisewski, Serkan Erdin, Eric Venner, Olivier Lichtarge, Robert Rentzsch, Haixuan Yang, Alfonso E Romero, Prajwal Bhat, Alberto Paccanaro, Tobias Hamp, Rebecca Kaßner, Stefan Seemayer, Esmeralda Vicedo, Christian Schaefer, Dominik Achten, Florian Auer, Ariane Boehm, Tatjana Braun, Maximilian Hecht, Mark Heron, Peter Hönigschmid, Thomas A Hopf, Stefanie Kaufmann, Michael Kiening, Denis Krompass, Cedric Landerer, Yannick Mahlich, Manfred Roos, Jari Björne, Tapio Salakoski, Andrew Wong, Hagit Shatkay, Fanny Gatzmann, Ingolf Sommer, Mark N Wass, Michael J E Sternberg, Nives Škunca, Fran Supek, Matko Bošnjak, Panče Panov, Sašo Džeroski, Tomislav Šmuc, Yiannis A I Kourmpetis, Aalt D J van Dijk, Cajo J F ter Braak, Yuanpeng Zhou, Qingtian Gong, Xinran Dong, Weidong Tian, Marco Falda, Paolo Fontana, Enrico Lavezzo, Barbara Di Camillo, Stefano Toppo, Liang Lan, Nemanja Djuric, Yuhong Guo, Slobodan Vucetic, Amos Bairoch, Michal Linial, Patricia C Babbitt, Steven E Brenner, Christine Orengo, Burkhard Rost, Sean D Mooney, and Iddo Friedberg. A large-scale evaluation of computational protein function prediction. *Nature Methods*, 10(3):221–227, January 2013. doi: 10.1038/nmeth.2340. URL https://doi.org/10.1038/nmeth.2340. 1

[7] Philip A Romero and Frances H Arnold. Exploring protein fitness landscapes by directed evolution. *Nature reviews Molecular cell biology*, 10(12):866–876, 2009. 1

[8] R. Fox, A. Roy, S. Govindarajan, J. Minshull, C. Gustafsson, J. T. Jones, and R. Emig. Optimizing the search algorithm for protein engineering by directed evolution. *Protein Engineering Design and Selection*, 16(8):589–597, August 2003. doi: 10.1093/protein/gzg077. URL https://doi.org/10.1093/protein/gzg077. 1

[9] Kevin K Yang, Zachary Wu, and Frances H Arnold. Machine-learning-guided directed evolution for protein engineering. *Nature methods*, 16(8):687–694, 2019. 1

[10] Zachary Wu, SB Jennifer Kan, Russell D Lewis, Bruce J Wittmann, and Frances H Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18):8852–8858, 2019. 3.1

[11] Bruce J. Wittmann, Yisong Yue, and Frances H. Arnold. Machine learning-assisted directed evolution navigates a combinatorial epistatic fitness landscape with minimal screening burden. *bioRxiv*, 2020. doi: 10.1101/2020.12.04.408955. URL https://www.biorxiv.org/content/early/2020/12/04/2020.12.04.408955. 1

[12] Martin Steinegger and Johannes Söding. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nature biotechnology*, 35(11):1026–1028, 2017. 1, 3.3

[13] Martin Steinegger and Johannes Söding. Clustering huge protein sequence sets in linear time. *Nature communications*, 9(1):1–8, 2018. 1

[14] UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1): D480–D489, 2021. 1, 3.2

[15] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Wang Yu, 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. doi: 10.1109/tpami.2021.3095381. URL `https://doi.org/10.1109/tpami.2021.3095381`. 1, 3

[16] Joshua Meier, Roshan Rao, Robert Verkuil, Jason Liu, Tom Sercu, and Alexander Rives. Language models enable zero-shot prediction of the effects of mutations on protein function. *bioRxiv*, 2021. 1, 2, 3, 5, 5

[17] Surojit Biswas, Grigory Khimulya, Ethan C Alley, Kevin M Esvelt, and George M Church. Low-N protein engineering with data-efficient deep learning. *Nature Methods*, 18(4):389–396, 2021. 1

[18] Thomas A Hopf, John B Ingraham, Frank J Poelwijk, Charlotta PI Schärfe, Michael Springer, Chris Sander, and Debora S Marks. Mutation effects predicted from sequence co-variation. *Nature biotechnology*, 35(2):128–135, 2017. 1

[19] Adam J Riesselman, John B Ingraham, and Debora S Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature methods*, 15(10):816–822, 2018. 2, 3

[20] Jonathan Frazer, Pascal Notin, Mafalda Dias, Aidan Gomez, Kelly Brock, Yarin Gal, and Debora Marks. Large-scale clinical interpretation of genetic variants using evolutionary data and deep learning. *bioRxiv*, 2020.

[21] Jung-Eun Shin, Adam J Riesselman, Aaron W Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C Kruse, and Debora S Marks. Protein design and variant prediction using autoregressive generative models. *Nature communications*, 12(1):1–11, 2021. 1

[22] Andriy Kryshtafovych, Torsten Schwede, Maya Topf, Krzysztof Fidelis, and John Moult. Critical assessment of methods of protein structure prediction (CASP)–Round XIII. *Proteins: Structure, Function, and Bioinformatics*, 87(12):1011–1020, 2019. 2

[23] Mohammed AlQuraishi. ProteinNet: a standardized data set for machine learning of protein structure. *BMC bioinformatics*, 20(1):1–10, 2019. 2

[24] Masthead. *Proteins: Structure, Function, and Bioinformatics*, 23(3):fmi–fmi, 1995. doi: https://doi.org/10.1002/prot.340230301. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.340230301`. 2

[25] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, pages 1–11, 2021. 2

[26] Amir Shanehsazzadeh, David Belanger, and David Dohan. Is transfer learning necessary for protein landscape prediction? *arXiv preprint arXiv:2011.03443*, 2020. 1, 2

[27] Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, 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. 1, 3, 3, 5, 5

[28] Roshan Rao, Nicholas Bhattacharya, Neil Thomas, Yan Duan, Xi Chen, John Canny, Pieter Abbeel, and Yun S Song. Evaluating protein transfer learning with TAPE. *Advances in neural information processing systems*, 32:9689, 2019. 1, 2, 3

[29] Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure. In *International Conference on Learning Representations*, 2018. 1

[30] Amy X Lu, Haoran Zhang, Marzyeh Ghassemi, and Alan M Moses. Self-supervised contrastive learning of protein representations by mutual information maximization. *BioRxiv*, 2020. 1

[31] Chloe Hsu, Hunter Nisonoff, Clara Fannjiang, and Jennifer Listgarten. Combining evolutionary and assay-labelled data for protein fitness prediction. *bioRxiv*, 2021. 1

[32] Naihui Zhou, Yuxiang Jiang, Timothy R Bergquist, Alexandra J Lee, Balint Z Kacsoh, Alex W Crocker, Kimberley A Lewis, George Georghiou, Huy N Nguyen, Md Nafiz Hamid, 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. 2

[33] Christophe Dessimoz, Nives Škunca, and Paul D Thomas. CAFA and the open world of protein function predictions. *Trends in Genetics*, 29(11):609–610, 2013. 2

[34] Tristan Bepler and Bonnie Berger. Learning the protein language: Evolution, structure, and function. *Cell Systems*, 12(6):654–669.e3, June 2021. doi: 10.1016/j.cels.2021.05.017. URL https://doi.org/10.1016/j.cels.2021.05.017. 2, 3

[35] Vanessa E Gray, Ronald J Hause, Jens Luebeck, Jay Shendure, and Douglas M Fowler. Quantitative missense variant effect prediction using large-scale mutagenesis data. *Cell systems*, 6(1): 116–124, 2018. 2, 3

[36] Roshan Rao, Joshua Meier, Tom Sercu, Sergey Ovchinnikov, and Alexander Rives. Transformer protein language models are unsupervised structure learners. In *International Conference on Learning Representations*, 2020. 2

[37] 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(1), December 2019. doi: 10.1186/s12859-019-3220-8. URL https://doi.org/10.1186/s12859-019-3220-8. 2, 3

[38] Céline Marquet, Michael Heinzinger, Tobias Olenyi, Christian Dallago, Michael Bernhofer, Kyra Erckert, and Burkhard Rost. Embeddings from protein language models predict conservation and variant effects. 2021. 2

[39] Maria Littmann, Michael Heinzinger, Christian Dallago, Tobias Olenyi, and Burkhard Rost. Embeddings from deep learning transfer GO annotations beyond homology. *Scientific reports*, 11(1):1–14, 2021.

[40] Hannes Stärk, Christian Dallago, Michael Heinzinger, and Burkhard Rost. Light attention predicts protein location from the language of life. *bioRxiv*, 2021. 2

[41] Nicholas C Wu, Lei Dai, C Anders Olson, James O Lloyd-Smith, and Ren Sun. Adaptation in protein fitness landscapes is facilitated by indirect paths. *Elife*, 5:e16965, 2016. 1, 3, 3.1

[42] W. Trent Franks, Benjamin J. Wylie, Sara A. Stellfox, and Chad M. Rienstra. Backbone conformational constraints in a microcrystalline u-15n-labeled protein by 3d dipolar-shift solid-state nmr spectroscopy. *Journal of the American Chemical Society*, 128(10):3154–3155, 2006. doi: 10.1021/ja058292x. URL https://doi.org/10.1021/ja058292x. PMID: 16522090. 1

[43] Drew H Bryant, Ali Bashir, Sam Sinai, Nina K Jain, Pierce J Ogden, Patrick F Riley, George M Church, Lucy J Colwell, and Eric D Kelsic. Deep diversification of an AAV capsid protein by machine learning. *Nature Biotechnology*, 39(6):691–696, 2021. 1, 3, 3.2

[44] Ran Zhang, Lin Cao, Mengtian Cui, Zixian Sun, Mingxu Hu, Rouxuan Zhang, William Stuart, Xiaochu Zhao, Zirui Yang, Xueming Li, Yuna Sun, Shentao Li, Wei Ding, Zhiyong Lou, and Zihe Rao. Adeno-associated virus 2 bound to its cellular receptor AAVR. *Nature Microbiology*, 4(4):675–682, February 2019. doi: 10.1038/s41564-018-0356-7. URL https://doi.org/10.1038/s41564-018-0356-7. 1

[45] Anna Jarzab, Nils Kurzawa, Thomas Hopf, Matthias Moerch, Jana Zecha, Niels Leijten, Yangyang Bian, Eva Musiol, Melanie Maschberger, Gabriele Stoehr, et al. Meltome atlas–thermal proteome stability across the tree of life. *Nature methods*, 17(5):495–503, 2020. 1, 3, 3.3

[46] Peer Bork and Eugene V Koonin. Predicting functions from protein sequences—where are the bottlenecks? *Nature genetics*, 18(4):313–318, 1998. 3

[47] A Elisabeth Sauer-Eriksson, Gerard J Kleywegt, Mathias Uhlén, and T Alwyn Jones. Crystal structure of the C2 fragment of streptococcal protein G in complex with the Fc domain of human IgG. *Structure*, 3(3):265–278, 1995. 3.1

[48] U Sjöbring, L Björck, and W Kastern. Streptococcal protein G. Gene structure and protein binding properties. *Journal of Biological Chemistry*, 266(1):399–405, 1991. 3.1

[49] LH Vandenberghe, JM Wilson, and G Gao. Tailoring the AAV vector capsid for gene therapy. *Gene therapy*, 16(3):311–319, 2009. 3.2

[50] Hildegard Büning, Anke Huber, Liang Zhang, Nadja Meumann, and Ulrich Hacker. Engineering the AAV capsid to optimize vector–host-interactions. *Current opinion in pharmacology*, 24: 94–104, 2015. 3.2

[51] Christopher Barnes, Olivia Scheideler, and David Schaffer. Engineering the AAV capsid to evade immune responses. *Current opinion in biotechnology*, 60:99–103, 2019. 3.2

[52] Georgios Mikos, Weitong Chen, and Junghae Suh. Machine learning identification of capsid mutations to improve AAV production fitness. *bioRxiv*, 2021. 3.2

[53] Sam Sinai, Nina Jain, George M Church, and Eric D Kelsic. Generative AAV capsid diversification by latent interpolation. *bioRxiv*, 2021. 3.2

[54] Ryan Lauchli, Kersten S Rabe, Karolina Z Kalbarczyk, Amulya Tata, Thomas Heel, Rebekah Z Kitto, and Frances H Arnold. High-throughput screening for terpene-synthase-cyclization activity and directed evolution of a terpene synthase. *Angewandte Chemie International Edition*, 52(21):5571–5574, 2013. 3.3

[55] J. D. Bloom, S. T. Labthavikul, C. R. Otey, and F. H. Arnold. Protein stability promotes evolvability. *Proceedings of the National Academy of Sciences*, 103(15):5869–5874, March 2006. doi: 10.1073/pnas.0510098103. URL `https://doi.org/10.1073/pnas.0510098103`. 3.3

[56] Yoshiaki Nosoh and Takeshi Sekiguchi. Protein engineering for thermostability. *Trends in biotechnology*, 8(1):16–20, 1990. 3.3

[57] Yougen Li, D Allan Drummond, Andrew M Sawayama, Christopher D Snow, Jesse D Bloom, and Frances H Arnold. A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments. *Nature biotechnology*, 25(9):1051–1056, 2007.

[58] Pete Heinzelman, Christopher D Snow, Indira Wu, Catherine Nguyen, Alan Villalobos, Sridhar Govindarajan, Jeremy Minshull, and Frances H Arnold. A family of thermostable fungal cellulases created by structure-guided recombination. *Proceedings of the National Academy of Sciences*, 106(14):5610–5615, 2009. 3.3

[59] Margaux M Pinney, Daniel A Mokhtari, Eyal Akiva, Filip Yabukarski, David M Sanchez, Ruibin Liang, Tzanko Doukov, Todd J Martinez, Patricia C Babbitt, and Daniel Herschlag. Parallel molecular mechanisms for enzyme temperature adaptation. *Science*, 371(6533), 2021. 3.3

[60] F. Y. Wu. The potts model. *Rev. Mod. Phys.*, 54:235–268, Jan 1982. doi: 10.1103/RevModPhys.54.235. URL `https://link.aps.org/doi/10.1103/RevModPhys.54.235`. 4

[61] Faruck Morcos, Andrea Pagnani, Bryan Lunt, Arianna Bertolino, Debora S. Marks, Chris Sander, Riccardo Zecchina, José N. Onuchic, Terence Hwa, and Martin Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many protein families. *Proceedings of the National Academy of Sciences*, 108(49):E1293–E1301, 2011. ISSN 0027-8424. doi: 10.1073/pnas.1111471108. URL `https://www.pnas.org/content/108/49/E1293`. 4

[62] William P Russ, Matteo Figliuzzi, Christian Stocker, Pierre Barrat-Charlaix, Michael Socolich, Peter Kast, Donald Hilvert, Remi Monasson, Simona Cocco, Martin Weigt, et al. An evolution-based model for designing chorismate mutase enzymes. *Science*, 369(6502):440–445, 2020. 4

[63] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

`

Who holds the Copyright on a NeurIPS paper

According to U.S. Copyright Office's page **What is a Copyright.** When you create an original work you are the and hold the copyright, unless you have an agreement to transfered the copyright to a third party such as the work for.

Authors do not tranfer the copyright of their paper to NeurIPS, instead they grant NeurIPS a non-exclusive, pe paid, fully-assignable license to copy, distribute and publicly display all or part of the paper