

Similarity Metrics for Numerical Simulations using Deep Learning

Georg Kohl

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. Nassir Navab, Ph.D.

Prüfende der Dissertation:

1. Prof. Dr.-Ing. Nils Thuerey
2. Prof. Dr. Mathias Niepert

Die Dissertation wurde am 11.04.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 19.06.2024 angenommen.

ABSTRACT

Transport-based physical phenomena described by partial differential equations are commonly encountered across scientific and engineering disciplines, for example, in heat conduction, the behavior of weather and climate, or the propagation of electromagnetic waves. A fundamental ingredient for the computational simulation of such processes is the similarity assessment of spatially varying physical quantities. Established similarity metrics for discretized simulation fields perform comparisons via element-wise operations. Due to their computational efficiency and a lack of better alternatives, such metrics are commonly used, even though they are suboptimal, as connected spatial structures or patterns in the data are not considered. This dissertation proposes two methods to address these shortcomings by utilizing deep learning techniques to compute semantically meaningful similarities that respect large-scale patterns.

The first part of this work targets the comparison of scalar data from two-dimensional simulations: A fundamental methodology for the similarity assessment is established, where two high-dimensional data points are embedded and compared in the latent space of a Siamese neural network. The data to train this network stems from a controllable data generation environment based on simulations of different partial differential equations. Ground truth distances for simulation fields against a reference sample are determined via the magnitude of a perturbation to the initial conditions of the simulation. Furthermore, a correlation-based loss function is developed to utilize relative similarities during training. The generalization capabilities of the resulting metric are evaluated in detail on an extensive range of synthetic test data and three real-world data sets.

In addition to targeting vectorial data from volumetric simulations, the second part of this dissertation also addresses several limitations of the first approach: The data acquisition methods are formalized, and a similarity model based on entropy is derived. It improves ground truth distances by adjusting them according to the decorrelation speed of different physical processes, leading to more effective training with less refined training data. Furthermore, a convolutional multiscale neural network architecture is introduced, and the correlation loss function from the first work is analyzed. Once again, the generalization abilities of the metric are demonstrated on various test sets, and its invariance properties to rotation and scale operations are investigated.

To summarize, the proposed approaches improve upon established element-wise similarity definitions and generalize well to new data. Nevertheless, the discussed results also indicate that several aspects of the similarity assessment of simulation data remain an open question for future research, e.g., explainability limitations.

ZUSAMMENFASSUNG

Transportphänomene, die durch partielle Differentialgleichungen beschrieben werden, sind in wissenschaftlichen und technischen Disziplinen weit verbreitet, zum Beispiel bei der Wärmeleitung, dem Verhalten von Wetter und Klima oder der Ausbreitung elektromagnetischer Wellen. Eine grundlegende Voraussetzung für die computergestützte Simulation solcher Prozesse ist die Ähnlichkeitsbewertung räumlich variierender physikalischer Größen. Etablierte Ähnlichkeitsmetriken für diskretisierte Simulationsfelder führen Vergleiche über elementweise Operationen durch. Aufgrund ihrer Berechnungseffizienz und des Mangels an besseren Alternativen werden solche Metriken häufig verwendet, obwohl sie suboptimal sind, da zusammenhängende räumliche Strukturen oder Muster in den Daten nicht berücksichtigt werden. Diese Dissertation schlägt zwei Methoden vor, um solche Mängel zu beheben, indem Deep Learning verwendet wird, um semantisch sinnvolle Ähnlichkeiten zu berechnen, die großskalige Strukturen berücksichtigen.

Der erste Teil dieser Arbeit zielt auf den Vergleich von skalaren Daten aus zweidimensionalen Simulationen ab: Es wird eine grundlegende Methodik für die Ähnlichkeitsbewertung etabliert, bei der zwei hochdimensionale Datenpunkte im latenten Raum eines Siamesischen neuronalen Netzwerks eingebettet und verglichen werden. Die Daten zum Training dieses Netzwerks stammen aus einer kontrollierbaren Datengenerierungsumgebung, die auf Simulationen verschiedener partieller Differentialgleichungen basiert. Die Ground Truth Distanzen für Simulationsfelder zu einer Referenzprobe werden über die Größe einer Störung der Anfangsbedingungen der Simulation bestimmt. Darüber hinaus wird eine korrelationsbasierte Verlustfunktion entwickelt, um relative Ähnlichkeiten während des Trainings zu nutzen. Die Verallgemeinerungsfähigkeiten der resultierenden Metrik werden im Detail an einer Vielzahl von synthetischen Testdaten und drei realen Datensätzen evaluiert.

Zusätzlich zur Zielsetzung vektorieller Daten aus volumetrischen Simulationen werden im zweiten Teil dieser Dissertation auch mehrere Einschränkungen des ersten Ansatzes angegangen: Die Datenerfassungsmethoden werden formalisiert und ein Ähnlichkeitsmodell basierend auf Entropie wird abgeleitet. Es verbessert Ground Truth Distanzen, indem es sie entsprechend der Dekorrelationsgeschwindigkeit verschiedener physikalischer Prozesse anpasst, was zu einem effektiveren Training mit weniger verfeinerten Daten führt. Darüber hinaus wird eine multiskalige neuronale Netzwerkarchitektur vorgeschlagen und die korrelationsbasierte Verlustfunktion aus der ersten Arbeit wird analysiert. Erneut werden die Verallgemeinerungsfähigkeiten der Metrik an verschiedenen Testdaten demonstriert, und ihre Invarianzeigenschaften gegenüber Rotations- und Skalierungsoperationen werden untersucht.

Zusammenfassend verbessern die vorgeschlagenen Ansätze etablierte elementweise Ähnlichkeitsdefinitionen und lassen sich gut auf neue Daten verallgemeinern. Dennoch zeigen die diskutierten Ergebnisse auch, dass mehrere Aspekte der Ähnlichkeitsbewertung von Simulationsdaten eine offene Frage für zukünftige Forschung bleiben, zum Beispiel Einschränkungen in Sachen Erklärbarkeit.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to everyone who supported me during my time at TUM. First and foremost, I want to thank Nils Thuerey for his experience, advice, and supervision from my bachelor's thesis to my dissertation. He guided me through all stages of multiple theses and research projects, provided regular and detailed feedback, and taught me how to approach research as a whole.

I also highly appreciate the support from my collaborators and advisors, Kiwon Um and Liwei Chen. Their expertise and patience helped me deal with difficulties and taught me how to approach and understand unfamiliar concepts. In addition, I am fortunate to have Prof. Mathias Niepert and Prof. Nassir Navab as members of my thesis committee.

Furthermore, I am grateful to all my colleagues who made the department lively and welcoming. Special thanks to Björn, Nilam, and Erik for the scientific (and sometimes non-scientific) conversations in our shared offices. Moreover, I want to thank Benjamin, Brener, Felix, Hao, Lukas, Mario, Marton, Patrick, Philipp, Qiang, Rachel, Rene, Robin, Shuvayan, Steffen, Stephan, and You for thorough discussions on research topics, and Michael and Sebastian for organizing various group activities. I also enjoyed the engaging lunch break conversations with Alexander, Behdad, Christian, Christoph, Fatemeh, Han, Josef, Junpeng, Kevin, Ludwig, and Simon, and I am grateful to Susanne and Sebastian for their help with administrative and technical issues. Overall, I really appreciate working with such talented, helpful, and genuinely friendly co-workers.

Last but not least, I want to thank my wife, family, and friends for their continuous support during my studies. Your encouragement helped me to remain motivated during difficult and exhausting moments. Thank you very much!

CONTENTS

Abstract	iii
Zusammenfassung	iv
Acknowledgments	vi
Acronyms	ix
1 Introduction	1
1.1 Contributions	5
1.2 List of Publications	6
1.3 Outline	7
2 Fundamentals and Related Work	8
2.1 Deep Learning	8
2.2 Simulation of PDEs	10
2.3 Similarity Assessment	14
3 Similarity Assessment of Simulation Data	20
3.1 Metrics via Siamese Architectures	20
3.2 Data Acquisition and Training	21
3.3 Evaluations and Results	24
4 Applications of Metrics based on Deep Learning	25
4.1 Accuracy Evaluations	27
4.2 Differentiable Loss Functions	29
5 Outlook	34
6 Conclusion	40
7 Summary of Publications	41
A Learning Similarity Metrics for Numerical Simulations	41
B Learning Similarity Metrics for Volumetric Simulations with Multiscale CNNs	42
Bibliography	43

Paper A – Published Version	56
Paper A – Appendix	68
Paper A – Permission Letter for Publication	88
Paper B – Published Version	89
Paper B – Appendix	98
Paper B – Permission Letter for Publication	108

ACRONYMS

BNN	Bayesian neural network.
CFD	computational fluid dynamics.
CNN	convolutional neural network.
DDPM	denoising diffusion probabilistic model.
DNS	direct numerical simulation.
FID	Fréchet inception distance.
FNO	Fourier neural operator.
GAN	generative adversarial network.
GNN	graph neural network.
LBM	lattice Boltzmann method.
LES	large eddy simulation.
LLM	large language model.
LPIPS	learned perceptual image patch similarity.
LSiM	learned simulation metric.
LSTM	long short-term memory.
MAE	mean absolute error.
MLP	multilayer perceptron.
MSE	mean squared error.
PCC	Pearson correlation coefficient.
PDE	partial differential equation.
PINN	physics-informed neural network.
PSNR	peak signal-to-noise ratio.
RANS	Reynolds-averaged Navier-Stokes.
RNN	recurrent neural network.
ROM	reduced order model.
SGD	stochastic gradient descent.

SPH smoothed particle hydrodynamics.
SRCC Spearman's rank correlation coefficient.
SSIM structural similarity index measure.

VAE variational autoencoder.
VolSiM volumetric similarity metric.

1 Introduction

Making comparisons is a fundamental human trait that is constantly performed consciously and subconsciously. For example, recognizing faces is based on comparing new to familiar faces, spatial orientation relies on comparing landmarks, or unfamiliar situations require constant adjustment between expectations and the environment. Comparisons are similarly significant across disciplines in science and engineering to make relative statements about data. This dissertation investigates comparisons of different data in terms of their similarity. Mathematically, comparisons in terms of similarity are modeled via metrics, i.e., distance functions that return a scalar distance value or similarity score, given two potentially high-dimensional input data points. Higher distances or lower similarity scores correspond to a lower degree of similarity between the inputs, and a lower distance bound or an upper similarity bound correspond to identical inputs.

When the inputs to the metric are constructs with little complexity, like scalar values, determining similarity can be as simple as a difference operation followed by an absolute value. However, it is typically necessary for practical applications to assess the similarity of substantially more complex high-dimensional data, like images or fields from physical simulations. Such similarity comparisons are, for instance, used to determine how accurately new generation or simulation techniques follow established models, theories, observations, or measurements (Oberkampf et al. 2004). This dissertation focuses on the similarity assessment of physical processes, especially motion-based transport problems and fluid flows. These processes are commonly encountered across a range of scientific disciplines, for example, in atmospheric sciences (Jolliffe and Stephenson 2012; Rasp et al. 2020), biomedical applications (Olufsen et al. 2000), astrophysics (Dehghan and Shakeri 2008), engineering in the form of aerodynamics and turbulence (Lin et al. 1998; Moin and Mahesh 1998; Pope 2000), or combustion (Pitsch 2006). Partial differential equations (PDEs) are typically used to describe these phenomena. If there are no analytic methods to compute solutions to the underlying equations directly, which is common, solutions can be computed approximately via numerical solvers. Such solvers employ spatiotemporal discretizations to approximate continuous quantities; in the simplest case, grids or tensors where quantities are stored in equally distant spatiotemporal positions,

similar to images that consist of pixels. This leads to the question of how such complex data structures can be meaningfully compared in terms of their similarity.

Fundamentally, even high-dimensional data can be considered an element of a vector space. As such, any metric in the corresponding space is a valid similarity measure, for example, the metrics induced by the commonly used L^p norms. These metrics are the high-dimensional counterparts to the simple scalar comparison example above and operate by definition *element-wise*, i.e., they only consider pairwise comparisons of individual vector components and aggregate them. This process ignores larger patterns or structures in the data and can be suboptimal, as illustrated by a sequence of examples below.

Assume that a given reference data sample b should be compared to two other samples a and c in terms of similarity. For both pairs, we focus on three different variants of similarity:

- (i) *Element-wise similarity*: Distances as computed with an element-wise metric.
- (ii) *Perceptual similarity*: Similarity as determined by humans with normal vision.
- (iii) *Semantic similarity*: Similarity based on the semantic methodology used to create each sample.

First, we investigate the simple example of a checkerboard pattern consisting of 16 pixels in Fig. 1.1. Black pixels correspond to the lowest value in the data range, and white pixels have the highest value. Compared to the reference b in Fig. 1.1b, the first sample a in Fig. 1.1a features the same checkerboard pattern translated by a single pixel. Sample c in Fig. 1.1c contains the average value from the data range across its entire domain. It is easy to observe that an element-wise analysis results in the highest possible distance for the pair (b, a) , while the pair (b, c) results in a distance magnitude that is 50% lower. However, sample a is clearly more similar to the reference perceptually and semantically since the fundamental checkerboard structure is preserved.

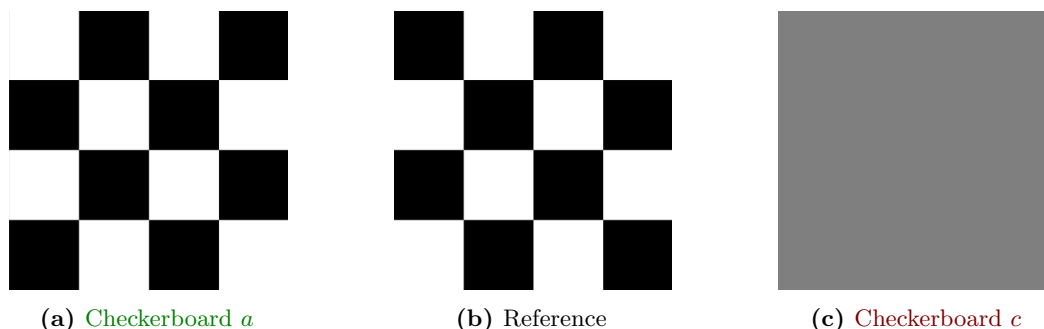


Figure 1.1: Checkerboard similarity example.

Figure 1.2 shows a similar problem with more realistic images: An element-wise evaluation results in a lower distance for the pair (b, c) , while (b, a) is typically preferred in a perceptual evaluation. Since image variants a and c rely on different transformations, determining a semantic similarity is fundamentally difficult in this case.

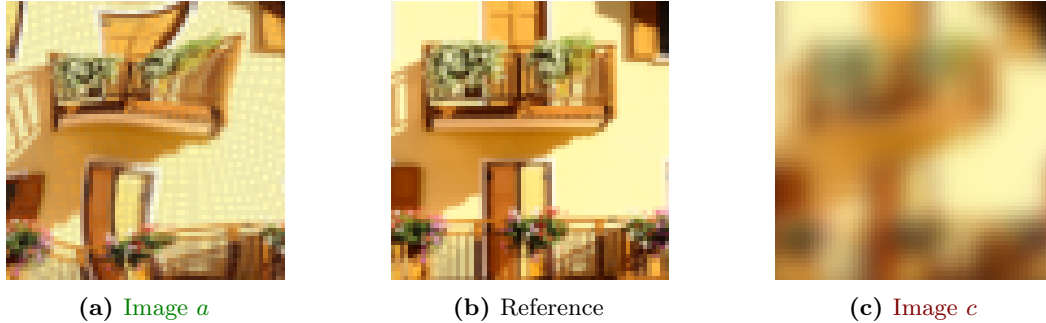


Figure 1.2: Image similarity example.¹

Establishing semantic similarities is easier for simulation data, as the generation process to create the data can be influenced. Figure 1.3 shows the density field extracted from a two-dimensional simulation of a buoyancy-driven rising smoke plume around an invisible force field directed downwards. Figure 1.3a and Fig. 1.3c are identical to the reference, apart from added noise with different variances to the velocity during the simulation. The noise variance for plume c had a larger magnitude compared to plume a , i.e., a more substantial effect on the simulation. As such, the semantic similarity is higher for the pair (b, a) due to a smaller difference in the initial condition of the simulations. This also mostly holds in a perceptual evaluation, while an element-wise metric predicts a lower distance for the pair (b, c) .

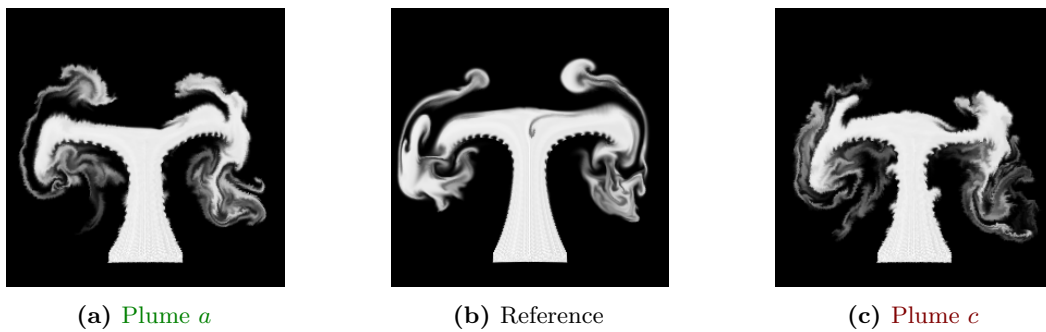


Figure 1.3: Plume similarity example in 2D.²

¹Images from Zhang et al. (2018).

²Simulation data from Kohl et al. (2020).

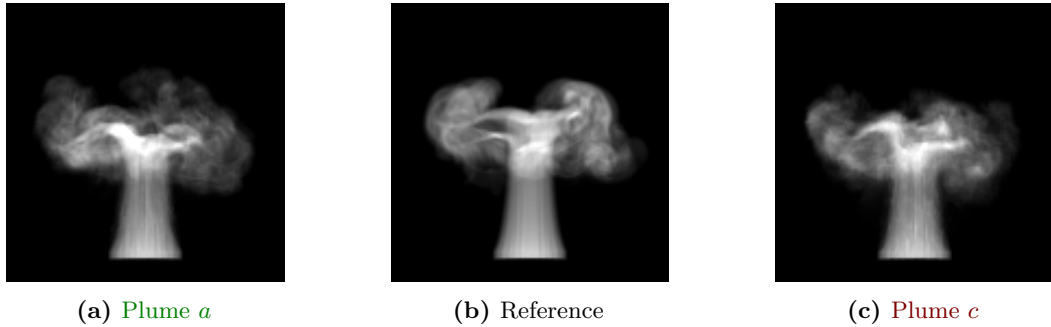


Figure 1.4: Plume similarity example in 3D, projected to 2D for visualization purposes.³

As a final example, [Fig. 1.4](#) shows another smoke plume from a similar setup, however, simulated in three dimensions and projected back to 2D for visualization purposes. In this case, the noise variance for plume c had twice the magnitude compared to plume a , and as a result, the semantic similarity is once again higher for the pair (b, a) . An element-wise metric predicts a lower distance for the pair (b, c) . The significant difference to the two-dimensional example is that meaningful perceptual evaluations are challenging due to the additional dimension: depending on the projection and method of displaying it, the perceptual similarity can change. While devices to display 3D data are conceivable, there is a fundamental limitation to the data dimensionality perceivable by humans.

Table 1.1: Summary of different similarities for the examples from [Figs. 1.1](#) to [1.4](#).

	Checkerboard	Image	Plume (2D)	Plume (3D)
Higher element-wise similarity	Checkerboard c	Image c	Plume c	Plume c
Higher perceptual similarity	clearly Checkerboard a	Image a	towards Plume a	highly unclear
Higher semantic similarity	Checkerboard a	unclear	Plume a	Plume a

To summarize, [Table 1.1](#) shows an overview of the examples above and illustrates three fundamental insights regarding the similarity assessment of data from numerical PDE simulations: First, element-wise metrics can be problematic in capturing perceptual or semantic similarities. That means commonly used point-wise comparisons based on L^p norms are suboptimal for meaningful comparisons of simulation data. Second, perceptual similarity is not extendable to high-dimensional simulation data. While visual evaluations might work for images and, to some degree, for low-dimensional simulations, they are not

³Simulation data from [Kohl et al. \(2023\)](#).

suitable for very complex high-dimensional data. Finally, semantic similarity can only be determined in a controlled environment. While the examples shown here are specifically designed to illustrate the mismatch between the results from element-wise metrics and the semantic context of the data, this does not generally work. Determining semantic similarity requires transformations that distort the data in a post-process or control over the data generation process, as in the plume examples. However, computing such a similarity measure is highly desirable, especially for numerical simulations. A typical use case is an accuracy evaluation of new simulation techniques against a known solution obtained, for example, from measurements or an expensive, highly accurate solver.

1.1 Contributions

In this dissertation, different methodologies to create metrics that follow a semantic similarity definition as outlined above are proposed and contextualized with respect to existing work. They operate on time-dependent two- or three-dimensional data from numerical simulations of various transport- and motion-based PDEs and are realized via deep learning techniques. The proposed methods rely on the following fundamental approach: First, a semantic similarity definition based on the simulation process that creates the data is established. Second, a large amount of training data with a corresponding semantic similarity score is generated using different solvers for various PDEs. Third, a metric function based on a neural network with a specialized architecture is trained on this training data. Finally, the neural network is validated on an extensive range of test data, and its properties are analyzed with different evaluations to ensure that it is reliable in practical applications. As shown below, the resulting metrics are superior to various commonly used element-wise vector space metrics and other deep learning-based image evaluation metrics. In addition, the proposed methods are analyzed with respect to their robustness, such as rotation or scale operations. Furthermore, this dissertation also includes a discussion of potential application domains using exemplary experiments from Kohl et al. (2024). The following summarizes the individual contributions of each work incorporated in this dissertation.

Learning Similarity Metrics for Numerical Simulations (Kohl et al. 2020)

This work presents the learned simulation metric (LSiM), a distance function for scalar, two-dimensional simulation data. The training data consists of simulations of the advection-diffusion equation, Burger’s equation, and the Navier-Stokes equations. The ground truth similarity is determined via perturbations to the initial conditions of the reference simulation, where the distance scales linearly with the perturbation strength.

Furthermore, a Siamese neural network architecture is derived from the mathematical properties of a pseudometric: First, a simulation pair is embedded into a high-dimensional latent space with a feature extractor network. The resulting feature maps are normalized, compared in the latent space via an element-wise distance, and aggregated to a scalar distance value with learned operations. The model is trained with a custom correlation loss function and evaluated on various generated out-of-distribution test sets and different real-world data. The central result of this work is that a specialized metric can capture semantic similarities better than established element-wise or image-based perceptual metrics. This work’s idea, methodology, and findings build upon the results from my master’s thesis (Kohl 2019).

Learning Similarity Metrics for Volumetric Simulations with Multiscale CNNs (Kohl et al. 2023) In this second follow-up work, the above methodology is substantially expended, in addition to considering vectorial, three-dimensional simulation data. First, an enhanced similarity model is derived from entropy that adjusts the ground truth distances to the reference simulation according to the decorrelation speed across different physical systems. Furthermore, two data generation schemes based on spatio-temporal coherences and perturbations to the initial conditions are formalized and automated. The importance of a suitable training data distribution is highlighted, and an improved multiscale neural network architecture is proposed. Furthermore, the tradeoff between a large batch size and an accurate correlation computation from the first work is investigated. The resulting volumetric similarity metric (VolSiM) is tested on a broad range of out-of-distribution test data, and its invariance to rotation and scale operations is evaluated. The central result from this work is that VolSiM is more suitable for analyzing high-dimensional simulation data than established element-wise and learned metrics. Furthermore, it generalizes well to substantially different simulations and is robust to input transformations.

1.2 List of Publications

The two core publications accompanying this publication-based dissertation were published as full papers at the peer-reviewed *International Conference on Machine Learning* (ICML 2020) and the *37th AAAI Conference on Artificial Intelligence* (AAAI 2023):

- **Paper A:** Georg Kohl, Kiwon Um, and Nils Thuerey (2020). "Learning Similarity Metrics for Numerical Simulations". In: *Proceedings of the 37th International*

Conference on Machine Learning (ICML 2020). Vol. 119, pp. 5349–5360. URL: <https://proceedings.mlr.press/v119/kohl20a.html>.

- **Paper B:** Georg Kohl, Li-Wei Chen, and Nils Thuerey (2023). "Learning Similarity Metrics for Volumetric Simulations with Multiscale CNNs". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37–7, pp. 8351–8359. DOI: 10.1609/aaai.v37i7.26007.

The published version of each paper is incorporated at the end of this dissertation: Paper A can be found on [Page 56](#), with the corresponding appendix on [Page 68](#), and Paper B begins on [Page 89](#), with the corresponding appendix on [Page 98](#). The data sets and source code accompanying both projects can be found at <https://github.com/tum-pbs/LSIM> and <https://github.com/tum-pbs/VOLSIM>, respectively.

Furthermore, additional results regarding the application of deep metrics are based on the following work, which is under review and only available as a preprint at the time of writing this dissertation:

- Georg Kohl, Li-Wei Chen, and Nils Thuerey (2024). *Benchmarking Autoregressive Conditional Diffusion Models for Turbulent Flow Simulation*. arXiv: 2309.01745 [cs, physics]. URL: <https://arxiv.org/abs/2309.01745>.

1.3 Outline

The main contents of this dissertation below are structured as follows: [Chapter 2](#) discusses relevant fundamental concepts and provides an overview of related research. [Chapter 3](#) describes the general proposed methodology for the similarity assessment of data from PDE simulations. Potential application domains where such a similarity assessment is beneficial are highlighted via exemplary experiments in [Chapter 4](#). [Chapter 5](#) critically reflects on the obtained results and provides an outlook for future work, and [Chapter 6](#) summarizes and concludes this dissertation. Finally, [Chapter 7](#) provides one-page summaries for each incorporated publication, including a brief overview of author contributions and a copyright statement.

2 Fundamentals and Related Work

This chapter provides an extensive overview of the fundamental concepts and related work on deep learning, simulation of PDEs, similarity assessment, and their overlap.

2.1 Deep Learning

Recently, deep learning has become a cornerstone in various scientific and engineering domains due to its ability to approximate arbitrary functions, given sufficient training data and certain theoretical restrictions (Cybenko 1989; Hornik et al. 1989). Over time, neural network architectures have evolved significantly, catering to diverse data types, domains, and learning tasks. For a more detailed introductory overview of deep learning, the reader is referred to Goodfellow et al. (2016) and Prince (2023).

Feed-forward Neural Networks Feed-forward neural networks, also known as multi-layer perceptrons (MLPs), consist of multiple neurons connected from the input to the output layer. Each neuron computes an affine transformation of its inputs with learnable weights and applies a non-linear activation function to the result. After a range of early experiments in the second half of the 20th century, the introduction of gradient-based optimization for the weights via backpropagation (Rumelhart et al. 1986) marked the beginning of deep learning. The work from Bengio et al. (2003) was among the first to utilize and explore MLPs, but they are still commonly used. MLPs introduce minimal prior knowledge about the data structure, making them ubiquitous but typically less performant than specialized architectures.

Convolutional Architectures Convolutional neural networks (CNNs) are designed to efficiently process data in regular Cartesian grids such as images while assuming local interactions and some degree of translation invariance (LeCun et al. 1989). They utilize convolutional layers, which apply a set of learnable filters to the input data, enabling hierarchical feature extraction. Early on, CNNs have achieved remarkable success in vision tasks like image classification (Krizhevsky et al. 2012; Simonyan and Zisserman

2015) or segmentation (Ronneberger et al. 2015), marking the begin of modern deep learning techniques. Recently, CNNs are an established architecture across tasks and domains and are common tools for learning with PDEs (see, e.g., Stachenfeld et al. 2022).

Graph Neural Networks Graph neural networks (GNNs) are tailored for data represented as graphs. They operate on the graph structure and node features, allowing them to capture relational information and dependencies within the data (Scarselli et al. 2009). GNNs have been applied to various tasks, including link prediction, graph classification, and node classification (Bronstein et al. 2017). They are ubiquitous for learning transport-based simulations, which often require particle-based representations or stretched, irregular, and unstructured grids (see, e.g., Sanchez-Gonzalez et al. 2020; Pfaff et al. 2021).

Recurrent Architectures Recurrent neural networks (RNNs) are specialized for sequential data, enabling them to capture temporal relations. Long short-term memorys (LSTMs) are an early RNN variant that addresses the vanishing gradient problem and enables learning of long-term dependencies (Hochreiter and Schmidhuber 1997). More recently, transformers (Vaswani et al. 2017) have been applied to sequence modeling and natural language processing, and they are the backbone of the recently emerging large language models (LLMs) like the GPT variants (Radford et al. 2019) or *LLama* (Touvron et al. 2023). In physical applications, recurrent architectures excel at learning time-dependent phenomena, typically in combination with reduced order models (ROMs) for the spatial dimensions (Wiewel et al. 2019; Han et al. 2021; Geneva and Zabaras 2022; Hemmasian and Farimani 2023).

Generative Models Generative models aim to learn the underlying probability distribution of the training data and generate new samples from it. Variational autoencoders (VAEs) learn a compressed latent representation of the data (Kingma and Welling 2014). At the same time, generative adversarial networks (GANs) consists of a generator and a discriminator model that compete against each other during a joint training (Goodfellow et al. 2014). Another recently emerging generative model class are diffusion models that learn to iteratively refine noise to samples from the data distribution (Ho et al. 2020; Song et al. 2021). They are the main driver behind recent image generation models like *Dall-E* (Ramesh et al. 2021) or *Imagen* (Saharia et al. 2022), and video generation models such as *Sora* (Brooks et al. 2024). Generative techniques are important in the context of

physics simulations for tasks like flow prediction or enhancing details via super-resolution (see, e.g., Xie et al. 2018; Kohl et al. 2024).

Training Procedures and Optimization Techniques Deep learning models are typically trained using stochastic gradient descent (SGD) and extensions like Adam (Kiefer and Wolfowitz 1952; Kingma and Ba 2015). Batch normalization and its variants, dropout, or different regularization methods are commonly employed to improve training stability and generalization performance (Srivastava et al. 2014; Ioffe and Szegedy 2015).

Transfer Learning and Pre-trained Models Transfer learning involves leveraging knowledge from pre-trained models on large datasets to improve performance on related tasks with limited data. CNN architectures pre-trained on ImageNet in the computer vision domain serve as starting points for various downstream learning tasks and applications (Deng et al. 2009; K. He et al. 2016). Such pre-trained feature extractors can be used via a Siamese network construction to create similarity measures that mimic human perception (Zhang et al. 2018). Style transfer is another area of application where the visual style from one learning task is combined with the semantic content of another domain. For instance, the resulting visually appealing artistic transformations can be used for images, videos, or fluid flows (Johnson et al. 2016; Ruder et al. 2016; B. Kim et al. 2019a).

2.2 Simulation of PDEs

PDEs are fundamental mathematical tools for modeling physical transport processes, which occur in many real-world phenomena such as heat conduction, fluid flow, and electromagnetic wave propagation (Evans 2022). This dissertation focuses on the following three PDEs: the advection-diffusion equation, Burgers' equation, and the full Navier-Stokes equations.

Transport-based PDEs The advection-diffusion equation in Eq. (2.1) describes the transport of a passive quantity in a flow, where u denotes the velocity, d is the scalar transported quantity, and ν is the diffusion coefficient or kinematic viscosity. The first term describes the dissipation of the passive quantity caused by the fluid's thermal conductivity and viscosity, i.e., diffusion, and the second term describes the transport of

the passive quantity along with the flow, i.e., advection.

$$\frac{\partial d}{\partial t} = \nu \nabla^2 d - u \cdot \nabla d \quad (2.1)$$

Similarly, the Burgers' equation in Eq. (2.2) describes how the flow's velocity field changes over time due to advection and diffusion. The diffusion term can also be interpreted as a viscosity that models the deformation resistance of the underlying material. Furthermore, this PDE can develop shock waves, i.e., discontinuities, making simulations more difficult due to numerical challenges.

$$\frac{\partial u}{\partial t} = \nu \nabla^2 u - u \cdot \nabla u \quad (2.2)$$

The incompressible Navier-Stokes equations describe the behavior of fluids like gases and liquids via modeling advection, viscosity, and pressure effects in Eq. (2.3), as well as mass conservation in Eq. (2.4). In addition to the notation above, P is the pressure, ρ is the density of the fluid, and f denotes external forces like gravity. The pressure at a given location corresponds to the force exerted by the mass of the surrounding fluid, and the conservation of mass in this formulation indicates that the fluid resists compression.

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{\nabla P}{\rho} + \nu \nabla^2 u + f \quad (2.3)$$

$$\nabla \cdot u = 0. \quad (2.4)$$

Traditional Simulation Techniques Traditional approaches for solving PDEs include finite difference (LeVeque 2007), finite element (Reddy 2019), and finite volume methods (Versteeg and Malalasekera 1995). These numerical techniques discretize the PDEs in space and time, allowing for the computation of approximate solutions with different strengths and weaknesses depending on the chosen simulation approach. While effective, traditional solvers may be computationally expensive and limited in their ability to handle complex geometries or boundary conditions. Common simulation frameworks in the domain of computational fluid dynamics (CFD) are open-source software like *SU2* (Economon et al. 2015) and *OpenFOAM* (Weller et al. 1998), as well as commercial products like *Simcenter STAR-CCM+* (2004) and *Ansys Fluent* (2006).

Depending on the requirements of the simulation, different techniques can resolve more or less details in the domain, leading to a trade-off between accuracy and computational requirements. Standard methods include direct numerical simulation (DNS) that fully resolves all frequencies of the flow, large eddy simulation (LES) that resolves large-scale turbulent structures explicitly while modeling small scales through a subgrid-scale model,

and Reynolds-averaged Navier-Stokes (RANS) simulations to produce a time-averaged result. The reader is referred to Pope (2000) for a detailed overview of different simulation methods and their mathematical background.

In computer graphics, fluids are typically simulated with more straightforward techniques, as physical accuracy is generally not required, and visual fidelity is sufficient. Examples are grid-based Eulerian approaches like the lattice Boltzmann method (LBM) from Frisch et al. (1986), particle-based Lagrangian techniques like smoothed particle hydrodynamics (SPH) from Monaghan (1992), or hybrid methods such as works from Stam (1999) and Zhu and Bridson (2005). To achieve real-time performance or further reduce computational costs of simulations, shallow water approaches or methods that add details in a post-process, such as the work from T. Kim et al. (2008), can be employed. For a more in-depth overview of fluid simulations in the context of computer graphics, the reader is referred to Bridson (2015).

Enhancing Solvers with Deep Learning Deep learning techniques can complement traditional solvers by accelerating convergence, improving stability, or enhancing accuracy. For example, a neural network-based method to solve the large, sparse linear systems encountered in a Eulerian simulation approach enabled the simulation of fluids in real-time (Tompson et al. 2017). A range of methods focus on improving under-resolved simulations with deep learning to achieve an accuracy that would traditionally be computationally very expensive at a lower cost. Examples include improved data discretizations via learned stencils (Bar-Sinai et al. 2019; Kochkov et al. 2021), or learned corrections to a low-resolution trajectory provided by a solver (Sirignano et al. 2020; Um et al. 2020; List et al. 2022). Geneva and Zabaras (2019) integrate uncertainty-aware deep learning approaches in RANS simulations to quantify model uncertainty. Furthermore, generative models like GANs or diffusion models can be used as a super-resolution post-process to increase visual fidelity or simulation accuracy without impacting the solver trajectory itself (Xie et al. 2018; Shu et al. 2023). For inverse problems like shape optimization or flow control, differentiable PDE solvers allow for including the solver into an optimization process to achieve predefined objectives (de Avila Belbute-Peres et al. 2020; Holl et al. 2020). Finally, deep learning techniques can also be used for artistic control over the resulting flow via style transfer approaches (B. Kim et al. 2019a, 2020). Thuerey et al. (2022) provide a detailed overview of different techniques for enhancing and replacing PDE solvers with deep learning.

Replacing Solvers with Deep Learning Recent research has also explored the use of deep learning to directly approximate solutions to PDEs in a fully data-driven manner,

bypassing the need for traditional solvers during inference. Simulations based on particles or with implicit representations typically require specialized learning methodologies that make use of such data structures: As an early example for learned particle-based simulations, the work from Ladicky et al. (2015) relies on regression forests for liquid simulations. More recently, approaches based on continuous convolutions were developed (Ummenhofer et al. 2020). Learning implicit representations of PDE solutions from data has been realized with methods such as physics-informed neural networks (PINNs) and Hamiltonian neural networks (Greydanus et al. 2019; Raissi et al. 2019).

There are two different paradigms to approaching simulations on regular meshes via deep learning: First, ROMs, which learn a compressed spatial representation that is evolved in time, and second, methods that operate directly on the input space. For the former, the reduced representations are typically achieved with architectures similar to autoencoders. The temporal latent model that evolves the resulting latent space forward in time can take different shapes. Early on, MLPs (B. Kim et al. 2019b) and LSTMs have been used successfully (Wiewel et al. 2019, 2020), but more recently transformer-based architectures are becoming increasingly common (Geneva and Zabarar 2022; Hemmasian and Farimani 2023).

Different techniques are viable for the latter method of working directly in the input space. Thuerey et al. (2020) replaced a RANS solver with a convolutional architecture to directly predict the steady-state flow around airfoils with different shapes. Li et al. (2021) proposed Fourier neural operators (FNOs) that transform the data to a spectral representation allowing for processing frequencies individually. Furthermore, it has been shown that even structurally simple CNN architectures can become powerful learned PDE simulators (Stachenfeld et al. 2022). Takamoto et al. (2022) and Gupta and Brandstetter (2023) provide various benchmark data sets of different PDEs and compare common neural network architectures for fully data-driven flow prediction. More recently, diffusion models were also applied as a replacement for traditional solvers, showing promising improvements in accuracy and temporal stability compared to earlier learned simulators (Lienen et al. 2023; Lippe et al. 2023; Kohl et al. 2024). Furthermore, curriculum-based training strategies and attention mechanisms initially designed for the transformer architecture have been used to improve the generalization abilities of autoregressive neural surrogate simulators to unseen PDE parameters (Takamoto et al. 2023).

Simulations on irregular meshes are commonly tackled with GNNs. A range of works relies on different message-passing or transformer-based architectures for the simulation of various PDEs (Sanchez-Gonzalez et al. 2020; Han et al. 2021; Pfaff et al. 2021; Brandstetter et al. 2022).

2.3 Similarity Assessment

A range of established metrics exist for the similarity assessment of PDE simulations and other high-dimensional data. The following provides an overview of traditional metrics and various deep learning methodologies that can be used for more meaningful comparisons across different domains.

Traditional Metrics Metrics induced by the L^p norms are highly prevalent for vector spaces. Depending on the value of p , they are known as Taxicab/Manhattan distance, Euclidean distance, Chebyshev distance, or Minkowski distance. In the context of comparisons for images or other higher-dimensional data, they are also referred to as mean absolute error (MAE), mean squared error (MSE), or peak signal-to-noise ratio (PSNR). These metrics provide quantitative measures of similarity or dissimilarity between data samples, and all have in common that they provide essentially element-wise comparisons that ignore larger structures or patterns in the data. For example, Aggarwal et al. (2001) provided theoretical insight on the usage of metrics based on L^p norms for high-dimensional data, and Huynh-Thu and Ghanbari (2008, 2012) investigated the limitations of the PSNR.

Other typical distance functions on vectors that boil down to element-wise metrics are the cosine similarity and correlation-based metrics like the Pearson correlation coefficient (PCC) or Spearman’s rank correlation coefficient (SRCC). The former measures the angle between two vectors and is particularly useful in text mining and recommendation systems. The latter indicate linear or monotonic relationships in two data samples, i.e., how changes in one variable affect the other (Spearman 1904; Pearson 1920). In information theory, variation of information indicates the information present in two sets of data and the information shared between them, and it is commonly used for data clustering (Meilă 2007). To compare probability distributions, the Wasserstein distance or Kantorovich–Rubinstein metric is a standard tool (Kantorovich 1960). It is related to the optimal transport problem, i.e., finding the most efficient way to transport a fixed amount of resources from one distribution to another while minimizing the total cost of transportation. As a result, it is also often referred to as the earth mover’s distance. Furthermore, the Kullback–Leibler divergence measures the differences between probability distributions more directly, without considering this transportation aspect (Kullback and Leibler 1951).

In the domain of images, a widely used metric for comparing the similarity between two images is the structural similarity index measure (SSIM). It is designed to assess the perceptual quality of images by considering their structural information, luminance, and

contrast (Z. Wang et al. 2003, 2004). This connection to the sensitivity towards the structural changes of the human visual system makes it more robust and reliable for image quality assessment than previously discussed metrics. Nevertheless, the formulation of SSIM has a direct link to the PSNR for specific image degradations (Horé and Ziou 2010). Furthermore, Nilsson and Akenine-Möller (2020) investigated a range of unexpected and undesirable behaviors of SSIM. While this analysis was performed in the context of rendered images in computer graphics, it still shows that SSIM exhibits similar issues like element-wise metrics and is not an ideal choice for the similarity assessment of data from numerical simulations.

Representation and Contrastive Learning Representation learning techniques, such as autoencoders and VAEs, seek to capture meaningful representations of data in a typically lower-dimensional space (for an overview, see Bengio et al. 2013). Such representation spaces are commonly achieved via unsupervised learning, for instance, from videos (Agrawal et al. 2015; Wang and Gupta 2015). While similarity assessments can be performed directly in the representation space, this is typically suboptimal as the space was not explicitly designed with similarity in mind. Instead, contrastive learning methods aim to learn unsupervised representations by contrasting positive pairs, i.e., similar samples, against negative pairs, i.e., dissimilar samples. Early applications include face verification or dimensionality reduction (Chopra et al. 2005; Hadsell et al. 2006). Typically, Siamese neural architectures are used for contrastive learning in the context of metrics. They consist of multiple sub-networks with shared weights, trained to learn embeddings that maximize similarity for similar inputs and minimize it for dissimilar inputs. Siamese architectures are widely used for similarity learning, one-shot learning, and metric learning (see, e.g., Koch et al. 2015), but also for tasks like object tracking (Bertinetto et al. 2016).

Invariances and Equivariances Invariance and equivariance properties are highly desirable in similarity metrics, as they ensure robustness to transformations such as translation, rotation, and scale. They enable meaningful comparisons across different instances of the same phenomenon. In addition to simple data augmentations that target an implicit learning process for equivariant and invariant representations, a range of research is also dedicated to specialized neural network architectures that learn such representations by design. Cohen and Welling (2016) proposed group equivariant networks for translations, reflections, and rotation operations, and similarly learned rotation equivariances were used for microscopy image analysis or volumetric data (Weiler et al. 2018; Chidester et al. 2019). In the domain of physics simulations, Cohen et al. (2018)

utilize CNNs for spherical images that commonly occur in weather and climate modeling, and R. Wang et al. (2021) incorporate equivariances to different transformations for improved generalization in the context of turbulent convection flows and ocean currents.

Perceptual Losses Perceptual losses are loss functions derived from deep neural networks trained for specific tasks, such as image classification or object detection. By incorporating such perceptual similarity measures into the loss function, models can be trained to generate outputs that are perceptually more similar to ground truth samples (Talebi and Milanfar 2018a). Training exclusively with element-wise losses can often lead to overly smooth, blurry results, while adding perceptual losses helps to improve the results in the high-frequency domain, like for highly textured areas (Dosovitskiy and Brox 2016). One explanation for this behavior is that CNNs trained on images tend to be biased towards the texture of objects rather than their shape (Geirhos et al. 2019). Perceptual loss formulations are commonly used in image-based tasks like style transfer and super-resolution (Johnson et al. 2016). Furthermore, losses based on learned metrics have been applied to semantic image compression (Mier et al. 2021) or video anomaly detection (Ramachandra et al. 2021).

Metrics based on Deep Learning Deep learning-based metrics allow for capturing high-level semantic features and perceptual differences by measuring similarity in the feature space of deep learning architectures. Early on, when deep learning emerged in the vision domain, a range of data sets and databases that feature human evaluations of images with different types and degrees of degradations and transformations were created (Larson and Chandler 2010; Liu et al. 2014; Ponomarenko et al. 2015). Typical early metric learning approaches directly learn these human evaluation results, for example, via Siamese CNN architectures (Amirshahi et al. 2016; Bosse et al. 2016; Kim and Lee 2017). It can be shown that the resulting metrics correspond to human perception of similarity, as, for instance, evaluated by Berardino et al. (2017) via the eigenvectors of the Fisher information matrix. Further improvements for this general methodology include predicting a distribution of human evaluations instead of the evaluations themselves (Prashnani et al. 2018; Talebi and Milanfar 2018b). There is a large body of research dedicated to similar metric learning methods for domains with different properties, which include voices and general audio signals (Zhang and Duan 2017; Avgoustinakis et al. 2020), interior and product design (Bell and Bala 2015), satellite image patch matching (H. He et al. 2019), textual and semantic similarity (Benajiba et al. 2019), and rendered images (Andersson et al. 2020).

In the context of training and evaluating probabilistic generative models, the inception score was proposed (Salimans et al. 2016). It combines an evaluation of the feature maps from the Inception architecture trained on ImageNet (Szegedy et al. 2015) with the Kullback–Leibler divergence. When employed as a similarity measure, it investigates the coverage and quality of the distributions generated by a model against the underlying ground truth data distribution. Heusel et al. (2017) improved upon this idea by using the Fréchet distance (a variant of Wasserstein distance) in conjunction with Inception model features, leading to the Fréchet inception distance (FID) commonly used to evaluate GANs and other generative models.

As an important milestone for deep metrics in the image domain, the seminal work from Zhang et al. (2018) investigated the features of a range of common ImageNet CNN architectures and different training modalities for their usage in perceptual image quality assessments. The authors found that starting from a pre-trained architecture like AlexNet (Krizhevsky et al. 2012) and fine-tuning the model with human perceptual evaluations lead to a highly accurate, versatile, and commonly used metric titled learned perceptual image patch similarity (LPIPS). However, they also report that random, untrained networks perform surprisingly well as perceptual metrics.

More recently, research on deep metrics either focuses on methodological and architectural improvements or investigates the limits of existing methods. For the former, self-supervised contrastive techniques were used to learn robust representations that work well as perceptual metrics, even without substantial fine-tuning (Madhusudana et al. 2022). In addition, Golestaneh et al. (2022) utilize a hybrid approach consisting of CNNs to learn local similarities and a transformer-based architecture to combine them for non-local interactions. Furthermore, a self-supervised self-consistency objective was proposed between an image and its equivariant transformation. Similarly, Lao et al. (2022) employed transformers to connect the similarity assessment of individual patches via CNNs by learning their relationships and contributions. To investigate the limitations of existing metrics and improve the general understanding of metric learning, Kumar et al. (2022) analyzed the relationship between ImageNet classification accuracy and performance as a metric, measured with a perceptual score. The authors found that modern vision architectures with medium to high ImageNet accuracy, including ResNets (K. He et al. 2016), EfficientNets (Tan and Le 2019), and Vision Transformers (Dosovitskiy et al. 2021) perform worse than simple CNNs as learned metrics. Furthermore, a critical bias in metrics trained on common data sets for image quality assessment was investigated by Ding et al. (2022): Metrics trained with perturbed images are overly sensitive to resampled textures. The authors propose a similarity model that combines structural and textural similarities to mitigate this shortcoming. Other problems in

terms of perturbation tolerance were revealed, indicating that learned metrics can suffer substantially from translation or scale operations of the inputs (Ghildyal and Liu 2022; Tsubota et al. 2022). Finally, various adversarial attacks on image metrics were documented by Kettunen et al. (2019) and Ghildyal and Liu (2023), indicating that most learned metrics still lack general robustness.

Similarity Assessment of PDE Simulations Assessing similarity between PDE simulations poses unique challenges due to the high-dimensional nature and complex structures and patterns in the data, but this remains an underexplored research direction. In weather forecasting, the displacement and amplitude score was an early approach to overcome the limitations of L^p norms by combining several heuristics with an optical flow-based distance function (Keil and Craig 2009). This distance is determined via the magnitude of a morphing velocity field required to transform one high-dimensional data point to the other. Such optical flows can be computed with traditional variational techniques or via deep learning (Horn and Schunck 1981; Ilg et al. 2017). In the first publication provided alongside this dissertation, an optical flow-based approach is investigated as a baseline metric as well (Kohl et al. 2020). Such methods can work well when changes are small but struggle once the fundamental assumption of optical flow is invalidated, i.e., that a particular locally confined structure or object moves between both inputs. For example, this can quickly happen when a vortex fully decays into fine scales or when two vortices merge into a larger one.

In the context of turbulent flows in CFD, frequency-based evaluation metrics are widespread (Pope 2000; Pitsch 2006). Such distance functions circumvent some issues of element-wise comparisons; for example, translated structures still exhibit similar spectral decompositions. However, their unstable behavior tends to be suboptimal when used as a comparison technique for individual data snapshots. Instead, they are ideal as a statistical tool when averaging spatially, temporally, or sample-wise over large amounts of data to achieve meaningful comparisons. This issue was also investigated by Kohl et al. (2024) in the context of evaluating diffusion models for turbulent flow prediction.

Siamese networks have been applied to retrieve fluid descriptors in a manner similar to metric learning approaches in the image domain. In their work, Chu and Thuerey (2017) pre-computed a repository of high-resolution flow snapshots that should be utilized to synthesize additional detail for a low-resolution simulation. A Siamese CNN architecture trained with a contrastive-style loss function is used to find a suitable descriptor from the repository. While such an approach is fundamentally suitable as an explicit similarity metric, the methods discussed in this dissertation improve upon a simple contrastive

setup with positive and negative pairs via more nuanced data samples with different degrees of dissimilarity from the reference.

Furthermore, crowd-sourced evaluations via the human visual system were proposed to assess simulations in the domain of physics-based animation and for the comparison of different discretization methods in the context of PDE simulations (Um et al. 2017, 2021). While their results show that perceptual evaluations for data from PDEs are possible, meaningful, and robust, several limitations are improved with the metrics based on deep learning discussed in this dissertation. First, visual evaluations require user studies that are slow and expensive. Second, they require careful consideration of formulations and questionnaire setup to avoid effects like framing, i.e., influences of the phrasing on the outcome of the question. Third, high-dimensional data can only be evaluated via projections to lower dimensions perceivable for humans, inevitably leading to similarity assessments influenced by the projection method, visualization techniques, or the loss of information due to the projection.

More recently, the comparison of distributions of turbulent flow fields has been tackled by Lienen et al. (2023). Like the FID, their work combines a Wasserstein metric with an underlying handcrafted distance function. First, the flow domain is separated into regions with similar velocity distributions via clustering. Afterward, a turbulence-specific distance for each area is derived from velocity and vorticity point statistics in the flow. The resulting distributional metrics are used to evaluate how well a predicted distribution of turbulent flow fields matches a known target distribution.

Evaluating the similarity of unsteady phenomena over time is another common but challenging topic. For example, in the context of predictions for chaotic systems, evaluation results of different methods against a known ground truth trajectory can substantially depend on the investigated temporal period when simply aggregating distances of individual temporal snapshots. While metrics going substantially beyond such straightforward aggregations remain an open problem, first steps have been taken by Gilpin (2021). This work analyzes the correlation between different temporal forecasting metrics and mathematical systems properties such as entropy, fractal dimension, and the Lyapunov exponent.

3 Similarity Assessment of Simulation Data

The following summarizes the general methodology of the two works incorporated in this dissertation. First, it is outlined how Siamese architectures can be utilized to compute distance functions suitable for the similarity assessment of PDE data. Afterward, the proposed semi-automatic data acquisition pipelines from different sources of simulation data and the training methodology are discussed. Finally, some key results from both works are highlighted.

3.1 Metrics via Siamese Architectures

The proposed metric architectures draw inspiration from perceptual metrics from the image domain, such as LPIPS (Zhang et al. 2018). They involve comparing network activations in the latent space of a feature extractor CNN instead of directly comparing data in the input space. A Siamese architecture is employed, i.e., both inputs undergo the same feature extraction process through a shared base network. Different network architectures for this feature extractor are viable. Still, the most robust results in our experiments were obtained with comparatively simple architectures like AlexNet (Krizhevsky et al. 2012) in two dimensions and a custom multiscale architecture for three-dimensional data. These findings align with recent research on images: Kumar et al. (2022) report the strong performance of relatively simple architectures compared to modern network variants closer to the state-of-the-art on other learning tasks.

The feature maps extracted by the Siamese network are normalized to address discrepancies in magnitude across different layers and features. Once again, different techniques were investigated in the first incorporated work, where a distribution-based normalization for the magnitude of the feature vectors was shown to work well. After the normalization, the latent spaces from both branches of the Siamese network corresponding to each input are compared via an element-wise comparison operation. The final step to achieve a metric is compressing the per-feature differences to a scalar distance value. Depending on the dimension, different learnable and non-learnable functions are typically employed. Combining different features should involve learned weights such that it is easy for the

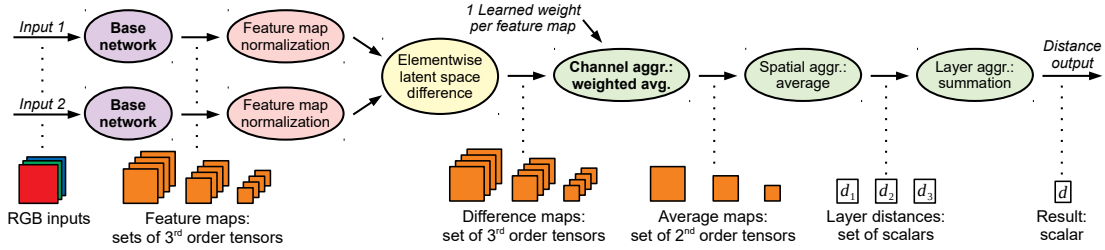


Figure 3.1: Siamese CNN architecture to compute a distance function from two input fields. Bold operations contain learned weights, and the shape of the results from each operation is illustrated at the bottom. In this example, inputs have two spatial dimensions, and the base network that extracts features consists of three layers with four feature maps each.⁴

optimization process to adjust the relative weight of the different features, especially when training with a frozen, pre-trained feature extractor. Figure 3.1 illustrates this process of transforming two inputs to a scalar distance value with a Siamese neural network approach. As shown in Kohl et al. (2020), this general methodology ensures the mathematical definition of a *pseudometric*. Compared to a full metric, only the identity of indiscernible is relaxed, i.e., a distance of zero for different inputs is allowed, while identical inputs have to receive a distance of zero.

3.2 Data Acquisition and Training

To train a metric for PDEs in a supervised manner, suitable training data has to be generated and equipped with the corresponding ground truth distances. Similar to training data for metrics in the image domain, the training data contains a reference state s_0 and variations s_1, s_2, \dots, s_n to that reference, with a decreasing degree of similarity. However, knowledge about the physical behavior of the underlying PDE is introduced instead of creating variations via simple transformations as a post-processing step.

Figure 3.2 illustrates two approaches to achieve this: either directly via the solver that is used to create the reference or with a spatiotemporal data repository if no solver is available. For the former, a reference simulation is run to create s_0 . Afterward, one parameter or initial condition p_i is varied by adding a perturbation Δ with an increasing factor $1, 2, \dots, n$ to create s_1, s_2, \dots, s_n . For the latter, a cutout at spatiotemporal position p is taken from a data repository as s_0 , and variants are created via additive spatiotemporal perturbations of magnitude Δ , once again scaled with increasing factors $1, 2, \dots, n$. For both methods, finding suitable magnitudes of Δ is difficult and requires

⁴Figure from Kohl et al. (2020).

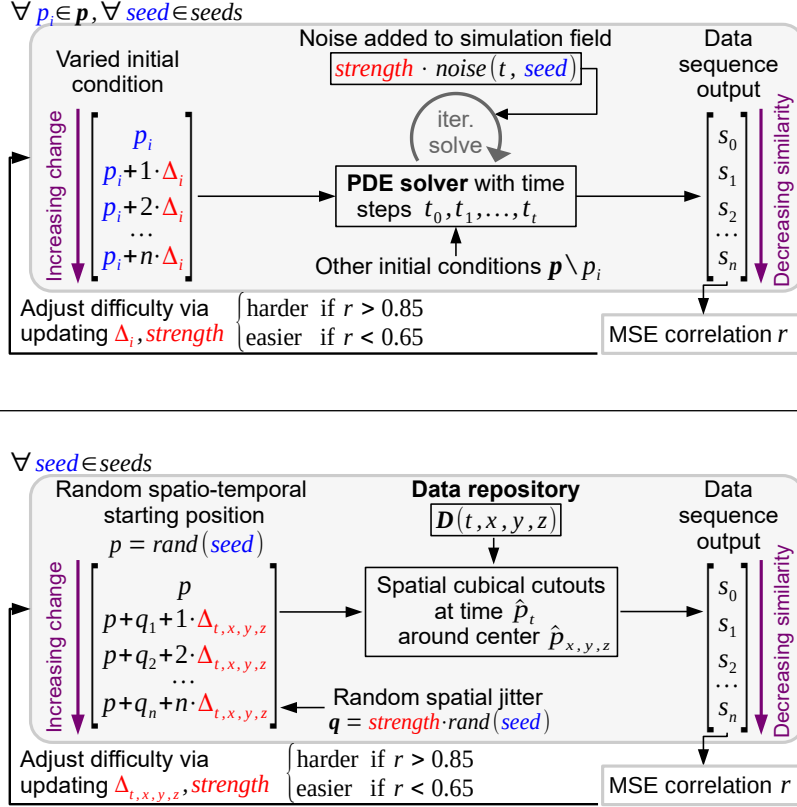


Figure 3.2: Data acquisition schemes to create and iteratively refine data in a controlled setup from a numerical solver (top) and a spatiotemporal data repository (bottom). For the former, increasingly different states from the reference are created via perturbations to the simulation’s initial conditions. For the latter, changes to the spatiotemporal cutout position are used.⁵

human intuition: When sequences change too little, they are too “easy”, i.e., do not show sufficiently interesting physical behavior for learning, but when changes are too big, they become too “hard” as barely any similarity is left between states. As such, the first incorporated work uses heuristics as a guide for adjustments, while the second work improves upon this via a correlation evaluation of an MSE proxy distance against the intended order s_0, s_1, \dots, s_n of the simulations, as shown in Fig. 3.2. Furthermore, the difficulty can be influenced by noise added to a field of the PDE simulation or with random spatial jitters when extracting data from a repository. The processes outlined above are repeated across different PDEs, initial conditions or cutout positions, and random seeds to generate complete data sets. Figure 3.3 shows an example from both data acquisition methods, consisting of the reference simulation and its perturbed variations.

⁵Figure from Kohl et al. (2023).

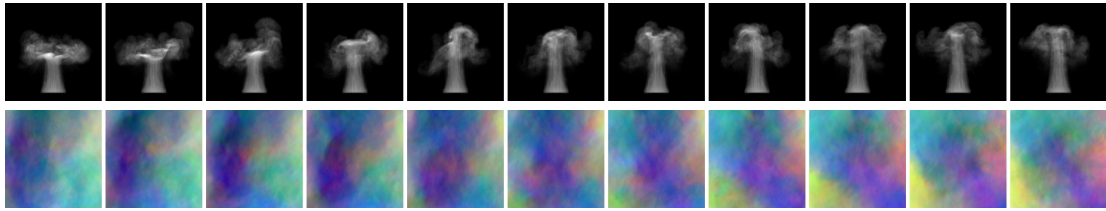


Figure 3.3: Volumetric example sequences corresponding to s_0, s_1, \dots, s_{10} projected to two dimensions for visualization purposes. The upper row contains the density fields of smoke plumes created via a simulation-based acquisition, where an invisible circular force field is at increasingly larger vertical positions from left to right. The lower row features isotropic turbulence at increasingly different spatiotemporal positions obtained with a repository-based acquisition.⁶

Next, ground truth distances for each state pair (s_i, s_j) where $i, j \in \{0, 1, \dots, n\}$ serve as the training target during optimization, as well as the desired solution during inference. The relative order of the state pairs is established by the data generation as outlined above, allowing for comparative statements like

$$g(s_0, s_1) < g(s_1, s_3) < g(s_1, s_6) < g(s_0, s_9), \quad (3.1)$$

where g denotes the distance function. Yet, defining absolute distance values for learning requires additional assumptions: As done in Kohl et al. (2020), the simplest solution is linearly distributing the distances in $[0, 1]$ via

$$g(s_i, s_j) = \frac{j - i}{n}. \quad (3.2)$$

However, additional information from the physical system can be used to account for different decorrelation speeds in different physical systems related to the size of Δ outlined above. As derived in more detail in Kohl et al. (2023), this can be achieved with

$$\tilde{g}(s_i, s_j) = \frac{\log\left(10^c \frac{j-i}{n} + 1\right)}{\log(10^c + 1)}, \quad (3.3)$$

where c is a system-dependent decorrelation speed. During training, directly comparing individual distance predictions d for a state pair from a neural network against g is a viable option. However, this does not explicitly reward fulfilling relative statements like Eq. (3.1). Thus, an additional loss term based on correlation is introduced in Kohl et al. (2020), which rewards distances that are correct relative to each other for all state pairs, even if the absolute values do not match the ground truth. Identical data augmentations like random flips, rotations, or crops on both states from a pair without altering the corresponding distances can increase the resulting metrics' robustness.

⁶Figure from Kohl et al. (2023) showing data extracted from a repository by Perlman et al. (2007).

3.3 Evaluations and Results

The most relevant evaluation for metrics based on deep learning is a generalization analysis on out-of-distribution data. For this purpose, both works incorporated in this dissertation evaluate all considered metrics on various such test sets. While most data and their ground truth distances are simulated or collected with the acquisition techniques above, they differ in the considered PDE or varied initial conditions. Furthermore, data from real-world scenarios and different domains is used. The former consists of multi-view reconstructions from real flows and measured weather data sets, while the latter features images, videos, or artificially translated shapes.

To measure the performance of different metrics, the SRCC or PCC of ground truth distances against predicted distances are utilized. The former only considers if the distances are ordered correctly relative to each other, while the latter measures linear relationships. Considering these correlation values, metrics based on deep learning consistently outperformed element-wise metrics across the investigated test sets, i.e., computed distances that come closer to the order dictated by the data generation. The PDE-specific metrics also improved upon specialized image metrics such as the SSIM or LPIPS, apart from image and video data sets. Furthermore, other learned distance function baselines were considered. They include a metric based on optical flow or a non-Siamese architecture that learns distances directly without relying on the inductive biases of mathematical metric properties. The former breaks easily when changes between inputs become too large or when the inherent assumptions of optical flow are violated. While the latter can learn distances surprisingly well, they lack robustness and generalization beyond the training domain.

Another critical analysis is the invariance to transformations like translation, rotation, and scale operations. Metrics should be fully invariant if both inputs are transformed with the same operation. Data augmentations provide some robustness, and the multiscale CNN architecture introduced in the second incorporated work helps further. Nevertheless, this is still an ongoing research direction, as discussed in more detail below and shown in related work in the image domain (Ghildyal and Liu 2022; Tsubota et al. 2022).

Finally, ablation studies help to understand deep metrics and motivate design decisions. In both works incorporated in this dissertation, a range of ablations on different aspects are performed, including neural network architecture components, different loss functions, configurations of the proposed loss, and training data difficulty and distribution.

4 Applications of Metrics based on Deep Learning

This chapter highlights different applications of deep learning-based metrics: their usage as accuracy evaluation tools and applications as differentiable loss functions during training. The presented data, models, and results are based upon the work from Kohl et al. (2024), which is under review at the time of writing this dissertation. While this work mainly investigates the usefulness of autoregressive conditional diffusion models for flow prediction, applications of the LSiM metric from the first publication incorporated in this dissertation are also showcased. The learning problem in the work from Kohl et al. (2024) consists of the fully data-driven prediction of temporally unsteady turbulent fluid flows given an initial condition. Different neural network architectures and training paradigms for this task were benchmarked on multiple experiments based on different fluid flows: First, an unsteady transonic cylinder flow with a Karman vortex street behind an immersed cylinder, and second, forced isotropic turbulence simulated via DNS.

The first experiment features compressible flows that exhibit complex shock waves in a transonic regime and are highly turbulent at a Reynolds number of 10^5 . During training, a set of flows with different Mach numbers Ma in the range $Ma \in [0.5, 0.9]$ are used. During inference, the models are required to extrapolate to Mach numbers outside of the training domain (Tra_{ext}) and interpolate to unseen Mach numbers inside the training range for short temporal rollouts of about two vortex shedding periods (Tra_{int}) and longer temporal rollouts of eight vortex shedding periods (Tra_{long}).

The second experiment consists of predicting flows in different two-dimensional planes taken from three-dimensional isotropic turbulence from the Johns Hopkins Turbulence Database (Perlman et al. 2007). This learning problem is inherently complex due to its underdetermined nature, as the flow in a plane also depends on the motion outside the plane. During training, the neural network architectures receive a set of planes and are required to generalize to a test set with new plane locations (Iso).

Different learning methodologies were investigated for these experiments: Autoregressive conditional diffusion models ($ACDM$) are derived from denoising diffusion probabilistic models (DDPMs) and feature a U-Net architecture (based on Ronneberger et al. 2015) as a model backbone with modernizations from Ho et al. (2020). Similar to DDPMs,

they operate by iteratively denoising a random initialization to the target flow state via conditioning on the previous simulation snapshots. This temporal one-step diffusion model is then unrolled autoregressively to achieve arbitrarily long simulation rollouts during inference.

The backbone U-Net architecture of *ACDM* can also be used without the iterative DDPM paradigm by training it as a simple next-step predictor that receives a previous simulation state and directly predicts the next one (*U-Net*). Unrolled training (*U-Net_{ut}*) similar to the work from Sirignano et al. (2020) and Um et al. (2020) is also utilized. Such gradient propagation through longer state trajectories during training helps to increase stability during inference.

Next, common architectures for learning PDE solutions are evaluated, which include dilated ResNets (*ResNet*) from Stachenfeld et al. (2022) and Fourier neural operators (*FNO*) from Li et al. (2021). Both are trained as next-step predictors like *U-Net* and unrolled autoregressively during inference. The former utilizes skip connections proposed for ResNets (K. He et al. 2016), allowing for efficient information transfer between network layers and different dilations in the convolution kernels to model local and global interactions. The latter employs Fourier transformations for its features to treat individual spectral components separately.

Finally, latent-space transformer architectures (*TF*) are analyzed, which are based on a spatial autoencoder model with an additional latent transformer that iteratively evolves the latent representation over time. They are trained on longer simulation rollouts to provide suitable learning signals to the latent transformer. Still, transformer architectures can infer sequences from a single initial state: First, the input is encoded, then the latent model evolves the latent space, and afterward, the latent trajectory is transformed back to the input space via the decoder model.

The following section illustrates how metrics based on deep learning can help to evaluate the accuracy of the resulting model trajectories for the experiments outlined above. Furthermore, it is shown that utilizing such metrics as differentiable loss functions during training can have benefits compared to training exclusively with element-wise comparisons. Unless denoted otherwise, the mean performance over all sequences from each data set and multiple training runs are reported below. For **Iso**, two training runs with different random seeds are evaluated, and three runs from **Tra**. In addition, five random model evaluations are considered per trained model for the probabilistic *ACDM* method.

4.1 Accuracy Evaluations

The behavior of three different metrics is compared to show how metrics based on deep learning can be useful in evaluating the accuracy of the predicted model trajectories: MSE, Pearson’s distance based on the PCC, and LSiM are employed in this comparison on the Tra_{long} test set. Lower values indicate better reconstruction accuracy for all metrics, and errors are computed via an average across data set sequences, training runs, probabilistic model evaluations, and simulation fields, i.e., velocity, pressure, and additionally density for Tra .

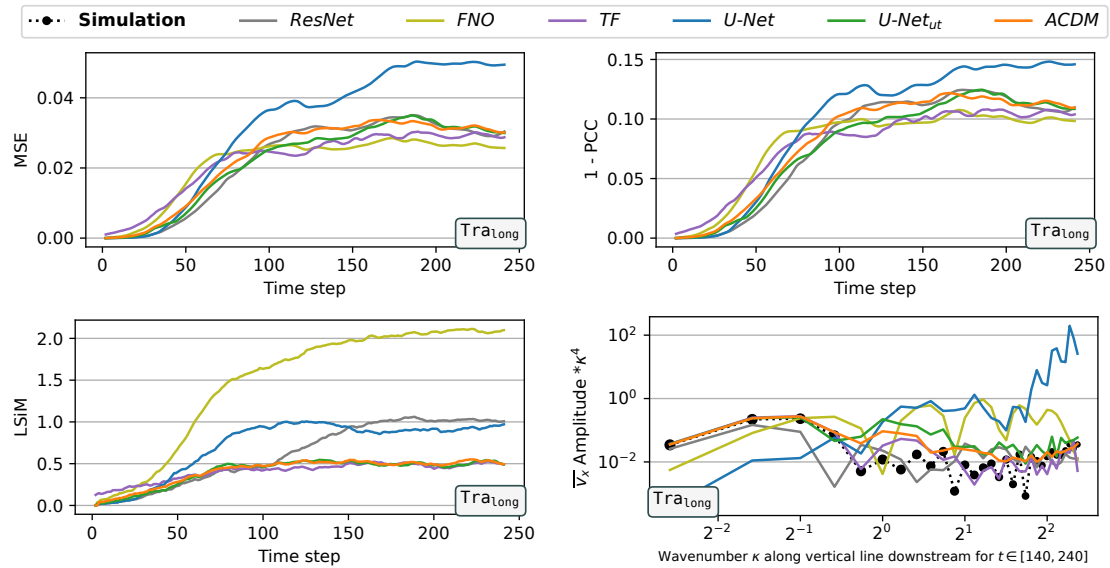


Figure 4.1: Error evolution over the temporal rollout on the Tra_{long} test set for different neural network architectures trained as surrogate models. Shown are evaluations with the MSE (upper left), Pearson’s distance (upper right), and LSiM (lower left). While the element-wise metrics at the top struggle to discern the quality of various predictions, LSiM separates FNO , $U\text{-Net}$, and $ResNet$, which perform differently compared to the remaining architectures. These differences are quantitatively measured via a spectral analysis of time steps $t \in [140, 240]$ (lower right) and qualitatively visualized in Fig. 4.2.

Figure 4.1 shows the error over time when evaluating the different architectures with the three similarity measures. Across evaluations, $U\text{-Net}$ is among the worst-performing architectures. MSE and Pearson’s distance behave pretty similarly regarding the relative performance of the different models. Note that there is a substantial difference between LSiM and the other two element-wise metrics: While the former can separate the temporal error of $ResNet$ and FNO from the remaining architectures, the latter attest a similar behavior to all models after time step $t = 100$. The physical differences between these predictions can be quantified with a spectral evaluation along a line downstream of the

immersed cylinder for $t \in [140, 240]$ shown in Fig. 4.1 at the lower right. The predictions of *ResNet*, *FNO*, and *U-Net* exhibit a different physical behavior than the remaining models indicated by differences from the simulation in the frequency spectrum.

To qualitatively compare the evaluation metrics, their match with a perceptual analysis of an exemplary trajectory can be utilized. Figure 4.2 displays representative visualizations for the prediction of one sequence from Tra_{long} . After about $t \approx 100$, the model predictions are fully decorrelated from the simulation trajectory, but there is a clear qualitative difference regarding the failure modes later on. While *U-Net* produces high-frequency artifacts, *ResNet* and *FNO* result in an incorrect mean flow prediction without vortices. Only *TF*, *U-Net_{ut}*, and *ACDM* keep a physical vortex-shedding behavior throughout the entire rollout. MSE and Pearson’s distance fail to capture this difference, as, from an element-wise perspective, a pure mean flow might even be closer to the simulation trajectory than a decorrelated yet plausible trajectory with vortices in different positions. LSiM, on the other hand, clearly separates models with physically plausible, decorrelated predictions from models that fully deteriorate, indicating the benefits of a deep learning-based similarity assessment.

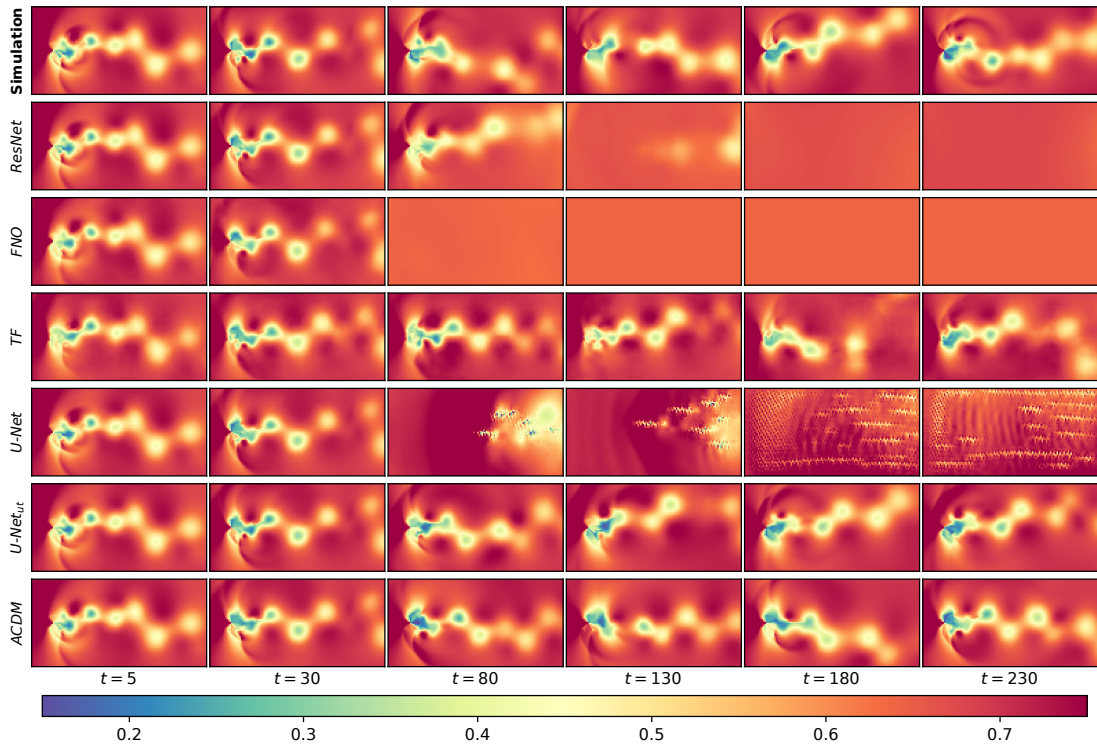


Figure 4.2: Pressure predictions for an example sequence from Tra_{long} across architectures.

4.2 Differentiable Loss Functions

In addition to accuracy evaluations, metrics based on deep learning can also be utilized as differentiable loss functions, similar to perceptual losses used for different learning tasks in the image domain (Dosovitskiy and Brox 2016; Johnson et al. 2016). Above, the *U-Net* model was trained solely with an element-wise MSE loss function, and employing LSiM as an additional loss term during training is investigated in more detail in the following. Given a predicted state s and a corresponding ground truth state \hat{s} , the training loss follows the form of

$$\mathcal{L}_{MSE+LSiM} = (s - \hat{s})^2 + \lambda * \text{LSiM}(s, \hat{s}). \quad (4.1)$$

In the following, the impact of λ , the weight that controls the influence of the LSiM loss, is investigated. *U-Net* models trained with e.g., $\lambda = 10^{-1}$, are denoted by *U-Net* $_{\lambda 1e-1}$. Note that $\lambda \approx 10^{-2}$ leads to a similar overall loss magnitude for the MSE and LSiM term during training. The resulting models are evaluated in terms of their accuracy, temporal stability, and spectral statistics on the out-of-distribution test sets to quantify the benefits of such training objectives.

Accuracy The predictions of the *U-Net* variants with different λ are evaluated via the MSE and LSiM to measure accuracy. Errors are computed via an average across data set sequences, training runs, time steps, and fields. Table 4.1 shows the accuracy on the test sets Tra_{ext} , Tra_{int} , and Tra_{long} from the transonic cylinder flow experiment, and *Iso* from the isotropic turbulence case. Errors of models that diverge during inference are displayed in gray with factors of 10^9 (b) in addition to the error scaling shown in the second table row. Choosing a suitable loss magnitude of around $\lambda = 10^{-3}$ substantially reduces errors in terms of LSiM across test sets. In addition, the added loss term consistently improves the prediction performance over the rollout in terms of the MSE. Note that the MSE errors on Tra_{long} for large values of λ do not follow the same pattern

Table 4.1: Accuracy evaluation for training *U-Net* with LSiM losses of different strength λ .

λ	Tra_{ext}		Tra_{int}		Tra_{long}		<i>Iso</i>	
	MSE (10^{-3})	LSiM (10^{-1})	MSE (10^{-3})	LSiM (10^{-1})	MSE (10^{-2})	LSiM (10^{-1})	MSE (10^{-2})	LSiM (10^{-1})
—	3.1 ± 2.1	3.9 ± 2.8	2.3 ± 2.0	3.3 ± 2.8	3.2 ± 0.7	8.9 ± 1.4	25.8 ± 35	11.3 ± 3.9
$1e-5$	4.2 ± 2.9	4.5 ± 3.0	2.6 ± 2.2	2.1 ± 2.0	4.6 ± 2.7	8.5 ± 2.9	67.4 ± 75.7	12.4 ± 3.8
$1e-4$	2.3 ± 1.2	3.7 ± 2.6	1.6 ± 1.4	2.0 ± 1.8	2.7 ± 0.6	7.8 ± 1.8	12.3 ± 9.3	11.8 ± 2.5
$1e-3$	2.9 ± 1.9	1.7 ± 0.8	2.2 ± 2.3	1.5 ± 0.9	2.7 ± 0.6	7.7 ± 2.1	6.3 ± 3.1	9.4 ± 2.8
$1e-2$	4.5 ± 1.3	3.5 ± 1.1	3.0 ± 2.3	1.8 ± 0.9	2.4 ± 0.2	10.0 ± 2.3	$0.1b \pm 0.2b$	15.3 ± 1.2
$1e-1$	5.8 ± 1.8	3.0 ± 0.8	5.2 ± 1.9	2.3 ± 0.6	2.9 ± 1.2	11.3 ± 0.7	$12b \pm 29b$	15.0 ± 1.0
$1e0$	6.8 ± 1.5	4.8 ± 1.1	6.6 ± 3.0	2.4 ± 0.7	2.5 ± 0.2	11.4 ± 1.0	$17b \pm 552b$	14.9 ± 1.0

as for all other cases. This is caused by the suboptimal behavior of element-wise metrics outlined in Section 4.1, and not reflected in the physical prediction quality as also visible in the example trajectories in Fig. 4.5. Adding minimal amounts of the LSiM loss term with $\lambda = 10^{-5}$ decreases performance. This behavior is most likely caused by the additional computations that lead to overly complex optimization landscapes and, thus, suboptimal gradient signals. Using excessively large factors for the loss term, such that it predominantly influences the overall loss magnitude, causes problems. Similarly, when training exclusively with loss metrics based on deep learning, the optimization process can get stuck in deteriorated solutions. During inference, this causes models to aggressively diverge after a few prediction steps, leading to substantial errors, especially on Iso as shown on the right in Table 4.1.

Temporal Stability The magnitude of the temporal rate of change is measured to assess the temporal stability of the different prediction trajectories. For states s^t and s^{t-1} at time t or $t - 1$, respectively, it is computed as

$$\left\| \frac{(s^t - s^{t-1})}{\Delta t} \right\|_1$$

for every normalized time step Δt . It indicates whether a surrogate simulator preserves the expected evolution of states as given by the reference simulation: If predictions collapse, the rate of change approaches zero, and if they explode, the rate of change grows substantially beyond the reference. Figure 4.3 shows this temporal stability evaluation averaged across data set sequences and training runs for the *U-Net* models on Tra_{long} and *Iso*. In line with the accuracy, *U-Net* $_{\lambda 1e-3}$ exhibits improved temporal stability compared to *U-Net* trained solely with an MSE loss. Choosing unsuitable λ causes models to diverge earlier from the reference trajectory when evaluating the difference between prediction steps for Tra_{long} and *Iso*.

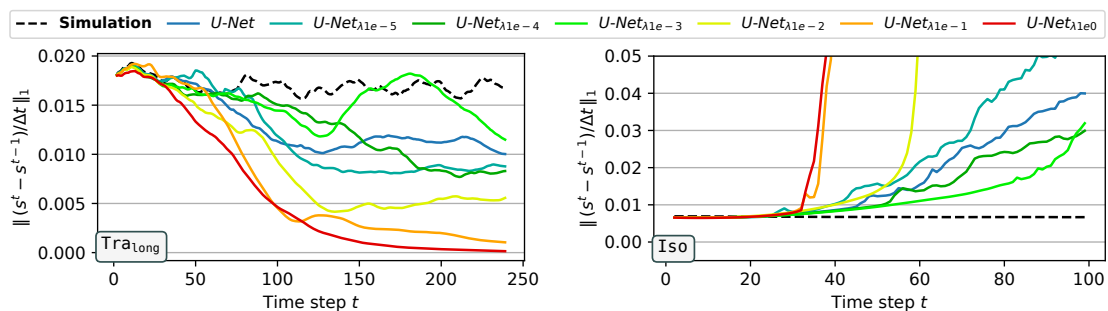


Figure 4.3: Temporal stability evaluation via the rate of change for *U-Net* models trained with LSiM losses of different strength λ on Tra_{long} (left) and *Iso* (right).

Spectral Behavior Evaluations for analyzing the spectral behavior of the models trained with LSiM are tailored to the investigated data set: For **Tra**, the amplitudes corresponding to different spatial wavenumbers of the horizontal motion across a vertical line in the flow are analyzed in an average over time. On **Iso**, the temporal frequency of the x-velocity is evaluated and averaged across every spatial point in the domain for a more stable analysis, as this case is isotropic. A single trajectory from the test set is employed in both cases, and the mean of the resulting spectra across trained models is used. As shown in Fig. 4.4, a spectral behavior closer to the simulation can be observed across the frequency band for $U\text{-Net}_{\lambda 1e-3}$ compared to $U\text{-Net}$ on both, Tra_{long} and **Iso**. Smaller values of λ still improve results on Tra_{long} but reduce spectral accuracy on **Iso**, while too large λ deteriorate predictions in both cases, with a more severe effect on **Iso**.

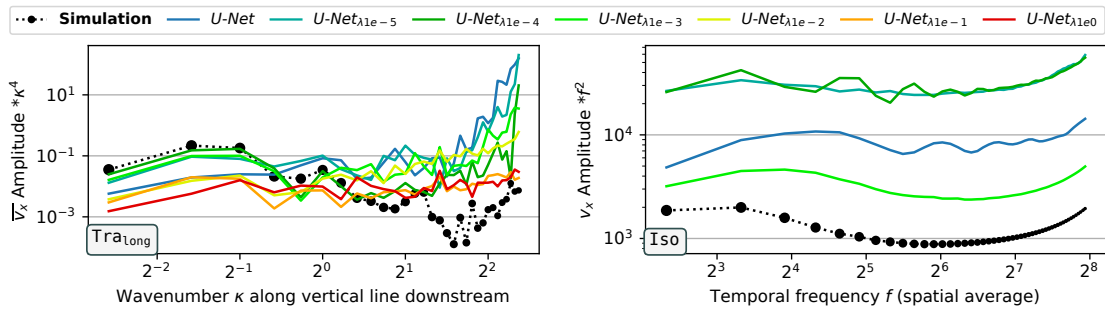


Figure 4.4: Spatial frequency analysis along a vertical line downstream on a sequence from Tra_{long} (left) and temporal frequency on a sequence from **Iso** (right) for models trained with LSiM losses of different strength λ . Missing models on **Iso** diverge during inference, resulting in frequency spectra that are substantially different from the simulation and are thus omitted.

To summarize, training with deep metrics as additional terms in the loss function has clear benefits regarding the underlying neural network architecture’s accuracy, temporal stability, and frequency behavior. This can also be confirmed qualitatively, as illustrated by the exemplary sequence visualizations of the pressure field from Tra_{long} in Fig. 4.5 and the vorticity from **Iso** in Fig. 4.6. Nevertheless, this added loss term requires a suitable choice for the loss weight hyperparameter, as the model performance might degrade otherwise.

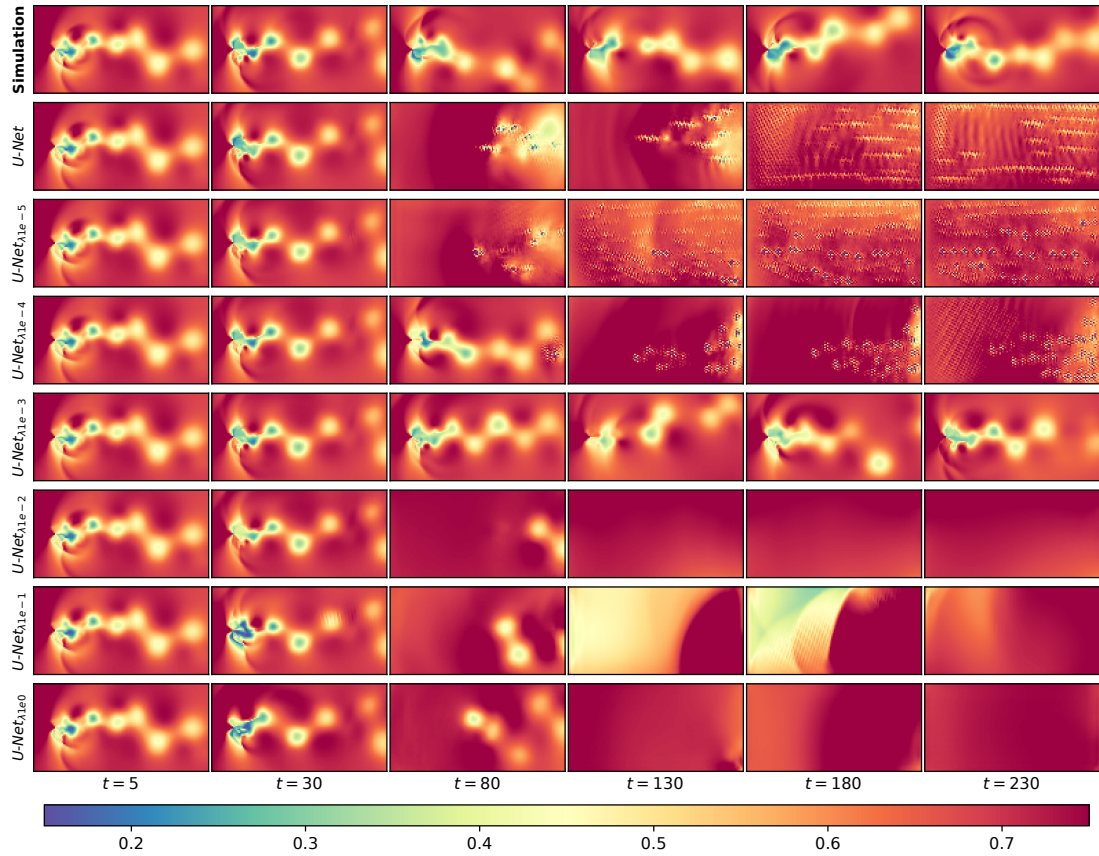


Figure 4.5: Pressure predictions for a sequence from $\text{Tra}_{1\text{ong}}$ for $U\text{-Net}$ models trained with an additional LSiM term weighted by different values of λ in the training objective.

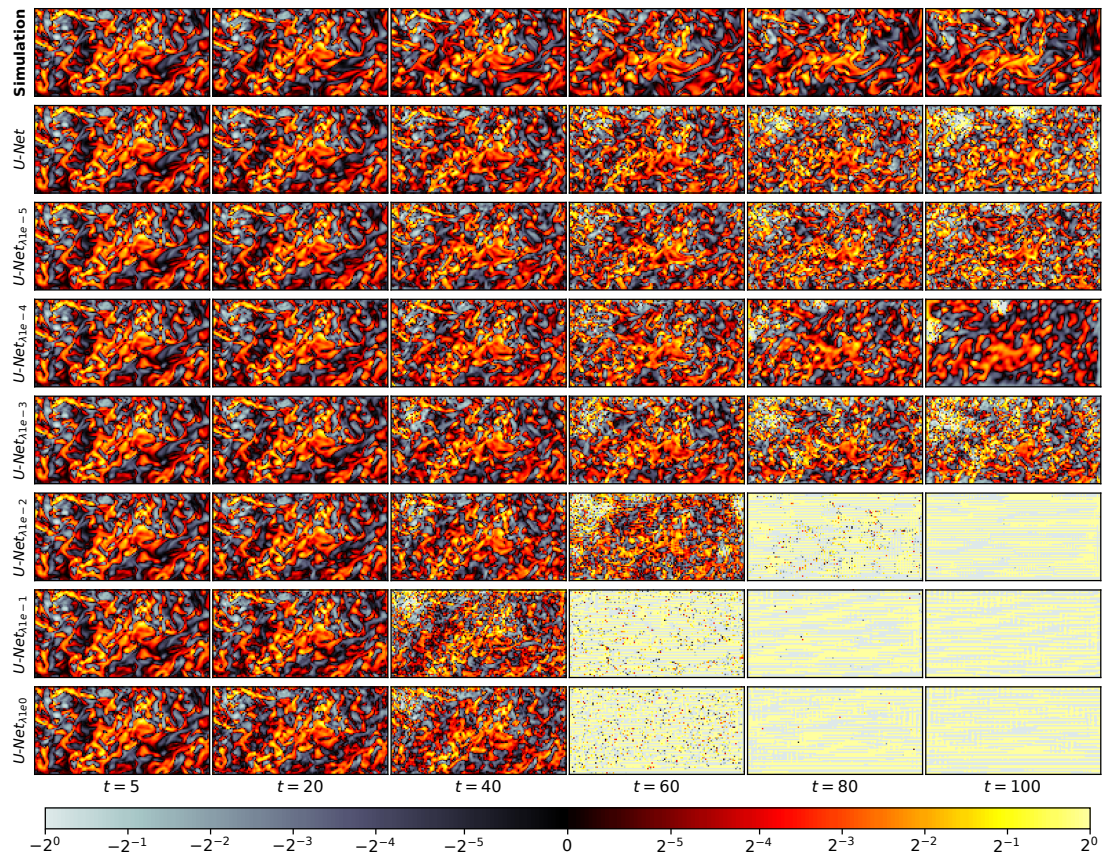


Figure 4.6: Vorticity predictions for a sequence from *Iso* for *U-Net* models trained with an additional LSiM term in the training objective weighted by different values of λ .

5 Outlook

There are two main directions future research in the domain of learned similarity metrics for PDE simulations could take besides the straightforward application as generic accuracy evaluation tools for various learning tasks. First, existing metric learning approaches can be leveraged for new applications and downstream learning tasks. In the image domain, learned perceptual metrics were successfully employed for downstream tasks like image generation (Dosovitskiy and Brox 2016), style transfer, super-resolution (Johnson et al. 2016), and more recently for semantic image compression (Mier et al. 2021), or to determine anomalies in videos (Ramachandra et al. 2021). Similarly, the methods proposed here can enhance a variety of downstream learning tasks on PDEs simulations as a loss function, as outlined in Section 4.2. Example learning tasks include fluid super-resolution (Xie et al. 2018), data-driven flow prediction (Kohl et al. 2024), or correction learning with solver integration (Um et al. 2020).

Second, several limitations of the proposed deep learning-based techniques remain that can be addressed in future work. The main areas for improvement are the learning approaches themselves, limitations concerning the data domain and modality, as well as interpretability and explainability.

Learning Approach As discussed in Chapter 3, the semi-automatic data generation techniques require human intuition to choose perturbed parameters and adjust their perturbation strength. This limits direct scalability to substantially more PDEs or initial conditions, but two potential directions can circumvent such limitations: First, integrating an additional optimization process that adjusts perturbation strengths is a step towards automating this process. However, such methods are computationally expensive and might not help to choose suitable parameters to vary. Second, employing unsupervised representation learning instead of explicit ground truth distances can remove the need for human intuition. Recently, self-supervised methods based on Lie symmetries have been applied to learning tasks with PDEs, but such representations are feasible as deep similarity measures with little adjustments (Mialon et al. 2023).

Similarly, Madhusudana et al. (2022) showed that self-supervised representations created via contrastive learning approaches are promising tools for metrics in the image domain. Fine-tuning learned representations with explicit supervision as a secondary training step is an option to increase performance with limited labeled data. As an additional benefit, self-supervised learning via invariant or equivariant representations as proposed by Cohen and Welling (2016), Weiler et al. (2018), and R. Wang et al. (2021) make learned metrics robust to different input transformations. In the image domain, it has been demonstrated that learned metrics are highly sensitive to operations like translations (Ghildyal and Liu 2022) or scaling (Tsubota et al. 2022). While first steps towards robust metrics for PDE data have been taken in the second work incorporated in this dissertation via data augmentations and a multiscale architecture, encoding such invariances directly as an inductive bias further increases reliability and theoretical guarantees.

Another limitation of the methods discussed in this dissertation is that fundamental assumptions to create ground truth distances are required since it is difficult to describe and formalize a desired similarity measure without already having a metric. Here, the assumption is that distances for the training data increase strictly monotonic with more substantial perturbations or spatiotemporal offsets. While this assumption can hold across many systems, it does not apply to certain classes and parameters, such as temporal offsets in a periodic vortex shedding flow behind a cylinder or similar oscillatory systems. The effects of using data from such problems for training are reasonably small when randomly sampling initial conditions and choosing limited perturbation magnitudes. Nevertheless, determining a suitable assumption to create distances for oscillatory cases or finding more general ground truth similarities that require no assumptions remain open problems.

Finally, different network architectures for metric learning have emerged recently in the image domain. Most notably, hybrid approaches between CNNs that learn mostly local features and transformers that learn global interactions and relationships have shown promising results (Golestaneh et al. 2022; Lao et al. 2022). Similar architectures are also worth investigating in the domain of PDE data in the future.

Domain Limitations While the proposed methods already investigate a broad range of transport and motion-based PDEs, adding further PDEs, simulation setups, perturbed parameters, perturbation amounts, and analyzed simulation fields can help to improve the robustness and generalization properties of the resulting metrics. While straightforward, such changes require substantial human intuition and additional tuning during the proposed semi-automatic data generation processes. As discussed above, fully unsupervised

representation learning techniques or methods requiring only small amounts of accurately calibrated data can make learned metrics more feasible on substantially larger scales.

An orthogonal direction is a deeper investigation of similarity assessments for individual PDEs or specific problem formulations. Examples include analyzing highly turbulent DNS simulations (Lin et al. 1998), metrics for particle-based simulations (Ummenhofer et al. 2020), or the similarity assessment of liquid surfaces (Um et al. 2017). Such specialized metrics can better consider prior, domain-specific knowledge to compute context-dependent similarities. For instance, they allow for controlling whether the metric should focus on localized phenomena such as turbulence dissipation or global phenomena like advection-diffusion problems and how the handling of boundary conditions or obstacles are reflected in the final distance.

As a final domain limitation, the proposed metrics are intended as instance comparisons, i.e., they predict distances given two flow fields, typically a predicted field and a known solution. Such comparisons are not very meaningful for generative tasks as generative models should create new samples following the training data distribution, not simply recreate individual samples. This is also an issue for chaotic systems when comparing trajectories beyond the chaotic timescale. One example of such systems is the highly chaotic Kuramoto–Sivashinsky equation (Sivashinsky 1977; Kuramoto 1978) where predictions quickly become decorrelated from a reference trajectory, and thus instance-wise comparison are not ideal. Instead, distributional metrics that statically compare if a distribution of generated samples or trajectories matches the target distribution are more suitable in such cases. Distributional comparisons are especially relevant to the recently emerging diffusion models that also find application in the fluid domain, e.g., for the probabilistic prediction of fluid flows (Cachay et al. 2023; Kohl et al. 2024). The most promising direction to achieve this is the combination of the proposed metrics with an established distribution similarity measure like the Kullback–Leibler divergence or a Wasserstein distance, similar as done by Salimans et al. (2016) and Heusel et al. (2017) for their inception score or FID. First steps towards such distributional comparisons of turbulent flows were taken by Lienen et al. (2023), who utilize an approach based on the 2-Wasserstein metric to compare a target data distribution to a distribution of probabilistic predictions. Their work uses a handcrafted underlying distance function to compare individual flows. It employs turbulence-specific point statistics of velocity and vorticity inside regions of similar flow behavior, which are determined via clustering. Future research can also target deep learning approaches to directly learn a distributional metric for PDE data. However, finding suitable training data and learning setups for this purpose remains an open problem.

Data Modality Limitations Both proposed methods are restrictive regarding the data modality due to the underlying CNN-based feature extractor architecture that prescribes a relatively rigid data dimensionality. Ideally, universal metrics for simulations should be able to handle data with an arbitrary number of spatial dimensions without relying on individual sub-models or simply resorting to slice-wise independent evaluations. While the VolSiM metric from the second discussed work was shown to be robust to input scale operations due to its fully convolutional multiscale architecture, this only holds within a limited range. When the spatial dimensions become very small, the downsampling operations prevent the computation of low-level features. Similarly, an enormous spatial scale can substantially reduce the effective receptive field of the network.

Furthermore, another modality limitation is the lack of support for time-dependent data. Analyzing complete temporal trajectories instead of individual snapshots is a widespread problem for unsteady phenomena; for example, for tasks like fluid flow prediction (Li et al. 2021; Geneva and Zabarar 2022; Stachenfeld et al. 2022; Kohl et al. 2024) or temporally coherent super-resolution of flows (Xie et al. 2018). Sliced temporal evaluations are possible with the proposed methods, yet they ignore temporal correlations and patterns, leading to similar issues as outlined for element-wise metrics in the spatial domain. First steps towards temporal metrics have been taken in the work from Gilpin (2021), where correlations between different forecast evaluation metrics and mathematical properties of physical systems are investigated. An extension of the proposed methods to time sequences can also be tackled with temporal convolution operations when suitable ground truth distances are utilized. Such operations have been applied to modeling temporal dynamics for flow prediction in work from R. Wang et al. (2020).

Finally and most importantly, the chosen CNN-based methodology eliminates a range of relevant, commonly used data discretizations across CFD or computer graphics applications: Particle-based simulations, irregular meshes, or fully unstructured domains cannot be evaluated directly (which are used in works from, e.g., de Avila Belbute-Peres et al. 2020; Ummenhofer et al. 2020; Pfaff et al. 2021). While it is possible to interpolate quantities to a regular grid for evaluation purposes, this introduces additional interpolation errors. In addition, it might require an enormous spatial grid resolution to accurately resolve all frequencies present in non-uniform data discretizations.

Explainability Explainability is a critical aspect of similarity assessment and deep learning approaches in general. Trained neural networks often act as black boxes and expose little insights regarding their inner workings to the user. This is also true for both methods investigated in this dissertation. While they adhere to the mathematical properties of pseudometrics as a theoretical grounding, further indications of how the

computed similarity scores are tied to the characteristics of the input pair are required. For example, it needs to be clarified which flow features of the data contribute in what manner to a low or high resulting distance. Potential directions to improve explainability include saliency maps (Simonyan et al. 2014), feature visualizations (Zeiler and Fergus 2014), or evaluation modes that report spatial distances instead of scalars (Zhang et al. 2018). Furthermore, analyzing if the resulting distances match perceptual evaluations of domain experts for some low-dimensional example scenarios can increase users' trust in deep learning-based metrics.

Explainable similarities are especially significant when applying learned metrics as differentiable loss functions. The lack of theoretical guarantees for outliers far from the training domain can be problematic in this context. As shown in Section 4.2, using high proportions of learned loss function components compared to element-wise losses can lead to performance deterioration. Training solely with a learned metric as the objective function even leads to training instability or models that do not substantially improve during training. This behavior is mainly caused by the optimization process of the downstream task getting stuck in the highly complex optimization landscape or finding loopholes in the metric definition. In the latter case, the optimizer reaches deteriorated solutions that result in low distances but do not solve the task in an intended way, i.e., there is a mismatch between the intended objective and the metric chosen in the loss function. Tackling such issues of metrics can reduce the required amount of parameter tuning when training with a conjunction of learned and element-wise loss terms or even eliminate the need for element-wise losses in the first place. Furthermore, training surrogate simulators exclusively with losses based on deep metrics can allow long training rollouts beyond a system's chaotic timescale, where element-wise loss formulations become problematic.

The limitations above resulting from training with deep metrics are also related to adversarial attacks (Goodfellow et al. 2015). Such attacks target the creation of input data with imperceptible perturbations specifically designed to fool a model into making incorrect predictions. Recent research in the image domain suggests that learned metrics can be highly susceptible to such attacks (Ghildyal and Liu 2023). Random transformation ensembles were proposed as possible techniques to mitigate this susceptibility (Kettunen et al. 2019). While adversarial attacks for PDE simulations are less crucial than in vision applications, achieving robust metrics is a relevant aspect for future work, nevertheless.

A similar direction targets a better understanding of learned metrics, as some unintuitive behaviors are not yet well understood. Zhang et al. (2018) report that random, untrained networks work surprisingly well as similarity metrics and the first work incorporated in this dissertation confirms this finding. Furthermore, Kumar et al. (2022) investigated the

relation between classification accuracy on ImageNet and performance as a metric via perceptual scores. The authors conclude that there is an inverse relationship between classification accuracy and perceptual scores in the medium to high-accuracy domain. This result indicates that progress on network architectures from established other domains might not translate well to learned deep metrics. Another unexpected finding was reported in work from Ding et al. (2022), who report that image metrics are overly sensitive when faced with resampled texture perturbations not encountered during training that are easily recognized to be similar by human evaluators. This can potentially be a widespread issue in the image domain, where it is common to train supervised metrics exclusively with a set of image perturbations stemming from a post-process. Similar problems can occur for the proposed metrics in this dissertation, as the training approach with perturbations does introduce a substantial bias that could have unexpected consequences.

Finally, another aspect highly relevant to explainability is uncertainty prediction. While generative architectures like GANs or VAEs allow for probabilistic sampling, explicit uncertainty predictions are commonly modeled via Bayesian neural networks (BNNs). BNNs are typically based on dropout layers that are active during inference (Srivastava et al. 2014) or on variational techniques (Blundell et al. 2015; Kingma et al. 2015). Kendall and Gal (2017) describe two types of uncertainty: aleatoric uncertainty inherent to the problem or the observations and epistemic uncertainty, which accounts for uncertainty in the model due to limited data. Creating similarity measures that predict both types of uncertainty is highly valuable. They allow for making statements about the quality of individual similarity assessments or can be used to prevent issues when employing learned metrics as loss functions in downstream optimization tasks. However, accurate uncertainty predictions for neural networks are still an open research question since such uncertainties are challenging to validate. First steps towards uncertainty-aware metrics in the image domain were taken by Prashnani et al. (2018) and Talebi and Milanfar (2018b) by learning similarity distributions for images created via many visual evaluations collected from human evaluators.

6 Conclusion

As illustrated above, the similarity assessment of data from numerical PDE simulations is a complex but highly relevant domain of ongoing research. Element-wise metrics cannot assess commonly occurring larger, structural changes in a semantically meaningful manner but remain the established evaluation criteria of choice across domains due to a lack of better alternatives. In this dissertation, different deep learning techniques were proposed, contextualized, and evaluated to compute such structure-aware metrics for data from PDEs. Furthermore, potential applications as accuracy evaluators or differentiable loss functions in the context of turbulent flow prediction were highlighted.

Meaningful and robust evaluation criteria are crucial as the overlap domain between machine learning and PDE simulations continues to evolve rapidly. Nevertheless, comparatively little research has been dedicated to analyzing the similarity of simulation data. The results presented in this dissertation demonstrate the importance of developing suitable evaluation methods alongside the simulation techniques themselves. We hope the introduced methodologies can provide insights and serve as starting points for future research on deep learning-based evaluation metrics for PDEs.

7 Summary of Publications

A Learning Similarity Metrics for Numerical Simulations

Abstract We propose a neural network-based approach that computes a stable and generalizing metric (*LSiM*) to compare data from a variety of numerical simulation sources. We focus on scalar time-dependent 2D data that commonly arises from motion and transport-based partial differential equations (PDEs). Our method employs a Siamese network architecture that is motivated by the mathematical properties of a metric. We leverage a controllable data generation setup with PDE solvers to create increasingly different outputs from a reference simulation in a controlled environment. A central component of our learned metric is a specialized loss function that introduces knowledge about the correlation between single data samples into the training process. To demonstrate that the proposed approach outperforms existing metrics for vector spaces and other learned, image-based metrics, we evaluate the different methods on a large range of test data. Additionally, we analyze generalization benefits of an adjustable training data difficulty and demonstrate the robustness of *LSiM* via an evaluation on three real-world data sets.

Author Contributions *Georg Kohl* contributed to the idea of the paper and was responsible for data acquisition, implementation of experiments, and evaluations. *Kiwon Um* provided feedback and conceptual insights and helped with user study evaluations during the development phase. *Nils Thuerey* guided the research project conceptually and contributed to high-level design decisions. The paper was written by *Georg Kohl* with substantial revisions and improvements from *Kiwon Um* and *Nils Thuerey*.

Copyright © 2020, The authors. No copyright was transferred to the publisher during the publication of this work, as detailed in the permission letter on [Page 88](#). The included reprint of the paper is identical to the version from Kohl et al. (2020) published in the Proceedings of Machine Learning Research (PMLR). Permission for reprinting was obtained from Kiwon Um and Nils Thuerey.

B Learning Similarity Metrics for Volumetric Simulations with Multiscale CNNs

Abstract Simulations that produce three-dimensional data are ubiquitous in science, ranging from fluid flows to plasma physics. We propose a similarity model based on entropy, which allows for the creation of physically meaningful ground truth distances for the similarity assessment of scalar and vectorial data, produced from transport and motion-based simulations. Utilizing two data acquisition methods derived from this model, we create collections of fields from numerical PDE solvers and existing simulation data repositories. Furthermore, a multiscale CNN architecture that computes a volumetric similarity metric (*VolSiM*) is proposed. To the best of our knowledge this is the first learning method inherently designed to address the challenges arising for the similarity assessment of high-dimensional simulation data. Additionally, the tradeoff between a large batch size and an accurate correlation computation for correlation-based loss functions is investigated, and the metric’s invariance with respect to rotation and scale operations is analyzed. Finally, the robustness and generalization of *VolSiM* is evaluated on a large range of test data, as well as a particularly challenging turbulence case study, that is close to potential real-world applications. (Kohl et al. 2023)

Author Contributions *Georg Kohl* was responsible for substantial parts of the idea for the paper and worked on the data generation, running the experiments, and implementing evaluations. *Li-Wei Chen* provided general feedback, contributed to the ground truth similarity model, and helped design some evaluations. *Nils Thuerey* guided the overall project direction and the conceptual design of experiments and evaluations. The paper was written by *Georg Kohl*, with minor revisions from *Li-Wei Chen* and substantial improvements from *Nils Thuerey*.

Copyright © 2023, Association for the Advancement of Artificial Intelligence (AAAI). Permission for the noncommercial reuse of all or portions of the above paper in other works of their own authorship is explicitly granted to the authors, as specified in the returned rights section of the permission letter on [Page 108](#). The content of the included paper reprint is identical to the version from Kohl et al. (2023) published by AAAI Publications. Permission for reprinting was obtained from Li-Wei Chen and Nils Thuerey.

BIBLIOGRAPHY

- Aggarwal, C. C., A. Hinneburg, and D. A. Keim (2001). “On the Surprising Behavior of Distance Metrics in High Dimensional Space”. In: *Database Theory - ICDT 2001*, pp. 420–434. ISBN: 978-3-540-44503-6. DOI: [10.1007/3-540-44503-X_27](https://doi.org/10.1007/3-540-44503-X_27).
- Agrawal, P., J. Carreira, and J. Malik (2015). “Learning to See by Moving”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 37–45. DOI: [10.1109/ICCV.2015.13](https://doi.org/10.1109/ICCV.2015.13).
- Amirshahi, S. A., M. Pedersen, and S. X. Yu (2016). “Image Quality Assessment by Comparing CNN Features between Images”. In: *Journal of Imaging Science and Technology* 60.6. DOI: [10.2352/J.ImagingSci.Technol.2016.60.6.060410](https://doi.org/10.2352/J.ImagingSci.Technol.2016.60.6.060410).
- Andersson, P., J. Nilsson, T. Akenine-Möller, M. Oskarsson, K. Åström, and M. D. Fairchild (2020). “FLIP: A Difference Evaluator for Alternating Images”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 3.2, 15:1–15:23. DOI: [10.1145/3406183](https://doi.org/10.1145/3406183).
- Ansys Fluent* (2006). Ansys Inc. URL: <https://www.ansys.com/products/fluids/ansys-fluent>.
- Avgoustinakis, P., G. Kordopatis-Zilos, S. Papadopoulos, A. L. Symeonidis, and I. Kompatsiaris (2020). “Audio-Based Near-Duplicate Video Retrieval with Audio Similarity Learning”. In: *25th International Conference on Pattern Recognition (ICPR 2020)*, pp. 5828–5835. DOI: [10.1109/ICPR48806.2021.9413056](https://doi.org/10.1109/ICPR48806.2021.9413056).
- de Avila Belbute-Peres, F., T. D. Economou, and J. Z. Kolter (2020). “Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*. Vol. 119, pp. 2402–2411. URL: <https://proceedings.mlr.press/v119/de-avila-belbute-peres20a.html>.
- Bar-Sinai, Y., S. Hoyer, J. Hickey, and M. P. Brenner (2019). “Learning Data Driven Discretizations for Partial Differential Equations”. In: *Proceedings of the National Academy of Sciences* 116.31, pp. 15344–15349. ISSN: 0027-8424, 1091-6490. DOI: [10.1073/pnas.1814058116](https://doi.org/10.1073/pnas.1814058116).
- Bell, S. and K. Bala (2015). “Learning Visual Similarity for Product Design with Convolutional Neural Networks”. In: *ACM Transactions on Graphics* 34.4, 98:1–98:10. DOI: [10.1145/2766959](https://doi.org/10.1145/2766959).
- Benajiba, Y., J. Sun, Y. Zhang, L. Jiang, Z. Weng, and O. Biran (2019). “Siamese Networks for Semantic Pattern Similarity”. In: *13rd IEEE International Conference on Semantic Computing (ICSC 2019)*, pp. 191–194. DOI: [10.1109/ICOSC.2019.8665512](https://doi.org/10.1109/ICOSC.2019.8665512).
- Bengio, Y., A. C. Courville, and P. Vincent (2013). “Representation Learning: A Review and New Perspectives”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8, pp. 1798–1828. DOI: [10.1109/TPAMI.2013.50](https://doi.org/10.1109/TPAMI.2013.50).

- Bengio, Y., R. Ducharme, P. Vincent, and C. Janvin (2003). “A Neural Probabilistic Language Model”. In: *Journal of Machine Learning Research* 3, pp. 1137–1155. URL: <http://jmlr.org/papers/v3/bengio03a.html>.
- Berardino, A., J. Balle, V. Laparra, and E. Simoncelli (2017). “Eigen-Distortions of Hierarchical Representations”. In: *Advances in Neural Information Processing Systems 30*. Vol. 30. URL: <http://arxiv.org/abs/1710.02266>.
- Bertinetto, L., J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr (2016). “Fully-Convolutional Siamese Networks for Object Tracking”. In: *Computer Vision - ECCV 2016 Workshops*. Vol. 9914, pp. 850–865. DOI: [10.1007/978-3-319-48881-3_56](https://doi.org/10.1007/978-3-319-48881-3_56).
- Blundell, C., J. Cornebise, K. Kavukcuoglu, and D. Wierstra (2015). “Weight Uncertainty in Neural Networks”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. Vol. 37. JMLR Workshop and Conference Proceedings, pp. 1613–1622. URL: <https://proceedings.mlr.press/v37/blundell115.html>.
- Bosse, S., D. Maniry, K.-R. Mueller, T. Wiegand, and W. Samek (2016). “Neural Network-Based Full-Reference Image Quality Assessment”. In: *2016 Picture Coding Symposium (PCS)*. DOI: [10.1109/PCS.2016.7906376](https://doi.org/10.1109/PCS.2016.7906376).
- Brandstetter, J., D. E. Worrall, and M. Welling (2022). “Message Passing Neural PDE Solvers”. In: *10th International Conference on Learning Representations (ICLR 2022)*. URL: <https://openreview.net/forum?id=vSix3HPYKSU>.
- Bridson, R. (2015). *Fluid Simulation for Computer Graphics*. CRC press. ISBN: 978-1-315-26600-8. URL: <https://doi.org/10.1201/9781315266008>.
- Bronstein, M. M., J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst (2017). “Geometric Deep Learning: Going beyond Euclidean Data”. In: *IEEE Signal Processing Magazine* 34.4, pp. 18–42. DOI: [10.1109/MSP.2017.2693418](https://doi.org/10.1109/MSP.2017.2693418).
- Brooks, T., B. Peebles, C. Homes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. Ng, R. Wang, and A. Ramesh (2024). “Video Generation Models as World Simulators”. In: *OpenAI*. URL: <https://openai.com/research/video-generation-models-as-world-simulators>.
- Cachay, S. R., B. Zhao, H. Joren, and R. Yu (2023). “DYffusion: A Dynamics-Informed Diffusion Model for Spatiotemporal Forecasting”. In: *Advances in Neural Information Processing Systems 36*. URL: http://papers.nips.cc/paper_files/paper/2023/hash/8df90a1440ce782d1f5607b7a38f2531-Abstract-Conference.html.
- Chidester, B., T. Zhou, M. N. Do, and J. Ma (2019). “Rotation Equivariant and Invariant Neural Networks for Microscopy Image Analysis”. In: *Bioinformatics* 35.14, pp. i530–i537. ISSN: 1367-4803. DOI: [10.1093/bioinformatics/btz353](https://doi.org/10.1093/bioinformatics/btz353).
- Chopra, S., R. Hadsell, and Y. LeCun (2005). “Learning a Similarity Metric Discriminatively, with Application to Face Verification”. In: *2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 539–546. DOI: [10.1109/CVPR.2005.202](https://doi.org/10.1109/CVPR.2005.202).
- Chu, M. and N. Thuerey (2017). “Data-Driven Synthesis of Smoke Flows with CNN-Based Feature Descriptors”. In: *ACM Transactions on Graphics* 36.4, 69:1–69:14. DOI: [10.1145/3072959.3073643](https://doi.org/10.1145/3072959.3073643).
- Cohen, T. and M. Welling (2016). “Group Equivariant Convolutional Networks”. In: *Proceedings of the 33rd International Conference on Machine Learning (ICML 2016)*. Vol. 48. JMLR

- Workshop and Conference Proceedings, pp. 2990–2999. URL: <https://proceedings.mlr.press/v48/cohenc16.html>.
- Cohen, T. S., M. Geiger, J. Köhler, and M. Welling (2018). “Spherical CNNs”. In: *6th International Conference on Learning Representations (ICLR 2018)*. URL: <https://openreview.net/forum?id=Hkbd5xZRb>.
- Cybenko, G. (1989). “Approximation by Superpositions of a Sigmoidal Function”. In: *Mathematics of Control, Signals, and Systems* 2.4, pp. 303–314. DOI: [10.1007/BF02551274](https://doi.org/10.1007/BF02551274).
- Dehghan, M. and F. Shakeri (2008). “Approximate Solution of a Differential Equation Arising in Astrophysics Using the Variational Iteration Method”. In: *New Astronomy* 13.1, pp. 53–59. ISSN: 1384-1076. DOI: [10.1016/j.newast.2007.06.012](https://doi.org/10.1016/j.newast.2007.06.012).
- Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). “ImageNet: A Large-Scale Hierarchical Image Database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).
- Ding, K., K. Ma, S. Wang, and E. P. Simoncelli (2022). “Image Quality Assessment: Unifying Structure and Texture Similarity”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.5, pp. 2567–2581. DOI: [10.1109/TPAMI.2020.3045810](https://doi.org/10.1109/TPAMI.2020.3045810).
- Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby (2021). “An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *9th International Conference on Learning Representations (ICLR 2021)*. URL: <https://openreview.net/forum?id=YicbFdNTTy>.
- Dosovitskiy, A. and T. Brox (2016). “Generating Images with Perceptual Similarity Metrics Based on Deep Networks”. In: *Advances in Neural Information Processing Systems* 29, pp. 658–666. URL: <https://proceedings.neurips.cc/paper/2016/hash/371bce7dc83817b7893bcdeed13799b5-Abstract.html>.
- Economou, T., F. Palacios, S. Copeland, T. Lukaczyk, and J. Alonso (2015). “SU2: An Open-Source Suite for Multiphysics Simulation and Design”. In: *AIAA Journal* 54, pp. 1–19. DOI: [10.2514/1.J053813](https://doi.org/10.2514/1.J053813).
- Evans, L. C. (2022). *Partial Differential Equations*. Vol. 19. American Mathematical Society. ISBN: 978-0-8218-4974-3.
- Frisch, U., B. Hasslacher, and Y. Pomeau (1986). “Lattice-Gas Automata for the Navier-Stokes Equation”. In: *Physical Review Letters* 56.14, pp. 1505–1508. DOI: [10.1103/PhysRevLett.56.1505](https://doi.org/10.1103/PhysRevLett.56.1505).
- Geirhos, R., P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel (2019). “ImageNet-Trained CNNs Are Biased towards Texture; Increasing Shape Bias Improves Accuracy and Robustness”. In: *7th International Conference on Learning Representations (ICLR 2019)*. URL: <https://openreview.net/forum?id=Bygh9j09KX>.
- Geneva, N. and N. Zabarar (2019). “Quantifying Model Form Uncertainty in Reynolds-Averaged Turbulence Models with Bayesian Deep Neural Networks”. In: *Journal of Computational Physics* 383, pp. 125–147. ISSN: 0021-9991. DOI: [10.1016/j.jcp.2019.01.021](https://doi.org/10.1016/j.jcp.2019.01.021).
- Geneva, N. and N. Zabarar (2022). “Transformers for Modeling Physical Systems”. In: *Neural Networks* 146, pp. 272–289. DOI: [10.1016/j.neunet.2021.11.022](https://doi.org/10.1016/j.neunet.2021.11.022).

- Ghildyal, A. and F. Liu (2022). “Shift-Tolerant Perceptual Similarity Metric”. In: *Computer Vision - ECCV 2022*. Vol. 13678. Lecture Notes in Computer Science, pp. 91–107. DOI: [10.1007/978-3-031-19797-0_6](https://doi.org/10.1007/978-3-031-19797-0_6).
- Ghildyal, A. and F. Liu (2023). “Attacking Perceptual Similarity Metrics”. In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: <https://openreview.net/forum?id=r9vGSpbBRO>.
- Gilpin, W. (2021). “Chaos as an Interpretable Benchmark for Forecasting and Data-Driven Modelling”. In: *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1*. URL: <https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/ec5decca5ed3d6b8079e2e7e7bacc9f2-Abstract-round2.html>.
- Golestaneh, S. A., S. Dadsetan, and K. M. Kitani (2022). “No-Reference Image Quality Assessment via Transformers, Relative Ranking, and Self-Consistency”. In: *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV 2022)*, pp. 3989–3999. DOI: [10.1109/WACV51458.2022.00404](https://doi.org/10.1109/WACV51458.2022.00404).
- Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. URL: <http://www.deeplearningbook.org>.
- Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio (2014). “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. URL: <https://proceedings.neurips.cc/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html>.
- Goodfellow, I., J. Shlens, and C. Szegedy (2015). “Explaining and Harnessing Adversarial Examples”. In: *3rd International Conference on Learning Representations (ICLR 2015)*. URL: <http://arxiv.org/abs/1412.6572>.
- Greydanus, S., M. Dzamba, and J. Yosinski (2019). “Hamiltonian Neural Networks”. In: *Advances in Neural Information Processing Systems 32*, pp. 15353–15363. URL: <https://proceedings.neurips.cc/paper/2019/hash/26cd8ecadce0d4efd6cc8a8725cbd1f8-Abstract.html>.
- Gupta, J. K. and J. Brandstetter (2023). “Towards Multi-Spatiotemporal-Scale Generalized PDE Modeling”. In: *Transactions on Machine Learning Research*. ISSN: 2835-8856. URL: <https://openreview.net/forum?id=dPSTDbGtBY>.
- Hadsell, R., S. Chopra, and Y. LeCun (2006). “Dimensionality Reduction by Learning an Invariant Mapping”. In: *2006 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2, pp. 1735–1742. DOI: [10.1109/CVPR.2006.100](https://doi.org/10.1109/CVPR.2006.100).
- Han, X., H. Gao, T. Pfaff, J.-X. Wang, and L. Liu (2021). “Predicting Physics in Mesh-Reduced Space with Temporal Attention”. In: *10th International Conference on Learning Representations (ICLR 2022)*. URL: <https://openreview.net/forum?id=XctLdNfCmP>.
- He, H., M. Chen, T. Chen, D. Li, and P. Cheng (2019). “Learning to Match Multitemporal Optical Satellite Images Using Multi-Support-Patches Siamese Networks”. In: *Remote Sensing Letters* 10.6, pp. 516–525. DOI: [10.1080/2150704X.2019.1577572](https://doi.org/10.1080/2150704X.2019.1577572).
- He, K., X. Zhang, S. Ren, and J. Sun (2016). “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: [10.1109/CVPR.2016.90](https://doi.org/10.1109/CVPR.2016.90).

- Hemmasian, A. P. and A. B. Farimani (2023). “Reduced-Order Modeling of Fluid Flows with Transformers”. In: *Physics of Fluids* 35.5, p. 057126. ISSN: 1070-6631. DOI: [10.1063/5.0151515](https://doi.org/10.1063/5.0151515).
- Heusel, M., H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter (2017). “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. In: *Advances in Neural Information Processing Systems 30*, pp. 6626–6637. URL: <https://proceedings.neurips.cc/paper/2017/hash/8a1d694707eb0fefe65871369074926d-Abstract.html>.
- Ho, J., A. Jain, and P. Abbeel (2020). “Denoising Diffusion Probabilistic Models”. In: *Advances in Neural Information Processing Systems 33*. URL: <https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html>.
- Hochreiter, S. and J. Schmidhuber (1997). “Long Short-Term Memory”. In: *Neural Computation* 9.8, pp. 1735–1780. DOI: [10.1162/NECO.1997.9.8.1735](https://doi.org/10.1162/NECO.1997.9.8.1735).
- Holl, P., N. Thuerey, and V. Koltun (2020). “Learning to Control PDEs with Differentiable Physics”. In: *8th International Conference on Learning Representations (ICLR 2020)*. URL: <https://openreview.net/forum?id=HyeSin4FPB>.
- Horé, A. and D. Ziou (2010). “Image Quality Metrics: PSNR vs. SSIM”. In: *20th International Conference on Pattern Recognition (ICPR 2010)*, pp. 2366–2369. DOI: [10.1109/ICPR.2010.579](https://doi.org/10.1109/ICPR.2010.579).
- Horn, B. K. and B. G. Schunck (1981). “Determining Optical Flow”. In: *Artificial intelligence* 17.1-3, pp. 185–203. DOI: [10.1016/0004-3702\(81\)90024-2](https://doi.org/10.1016/0004-3702(81)90024-2).
- Hornik, K., M. B. Stinchcombe, and H. White (1989). “Multilayer Feedforward Networks Are Universal Approximators”. In: *Neural Networks* 2.5, pp. 359–366. DOI: [10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8).
- Huynh-Thu, Q. and M. Ghanbari (2008). “Scope of Validity of PSNR in Image/Video Quality Assessment”. In: *Electronics Letters* 44.13, pp. 800–801. ISSN: 0013-5194. DOI: [10.1049/el:20080522](https://doi.org/10.1049/el:20080522).
- Huynh-Thu, Q. and M. Ghanbari (2012). “The Accuracy of PSNR in Predicting Video Quality for Different Video Scenes and Frame Rates”. In: *Telecommunication Systems* 49.1, pp. 35–48. ISSN: 1572-9451. DOI: [10.1007/s11235-010-9351-x](https://doi.org/10.1007/s11235-010-9351-x).
- Ilg, E., N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox (2017). “FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1647–1655. DOI: [10.1109/CVPR.2017.179](https://doi.org/10.1109/CVPR.2017.179).
- Ioffe, S. and C. Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. Vol. 37. JMLR Workshop and Conference Proceedings, pp. 448–456. URL: <https://proceedings.mlr.press/v37/ioffe15.html>.
- Johnson, J., A. Alahi, and L. Fei-Fei (2016). “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. In: *Computer Vision - ECCV 2016*. Vol. 9906, pp. 694–711. DOI: [10.1007/978-3-319-46475-6_43](https://doi.org/10.1007/978-3-319-46475-6_43).
- Jolliffe, I. T. and D. B. Stephenson (2012). *Forecast Verification: A Practitioner’s Guide in Atmospheric Science*. John Wiley & Sons. URL: <https://doi.org/10.1002/9781119960003>.

- Kantorovich, L. V. (1960). “Mathematical Methods of Organizing and Planning Production”. In: *Management science* 6.4, pp. 366–422. DOI: [10.1287/mnsc.6.4.366](https://doi.org/10.1287/mnsc.6.4.366).
- Keil, C. and G. C. Craig (2009). “A Displacement and Amplitude Score Employing an Optical Flow Technique”. In: *Weather and Forecasting* 24.5, pp. 1297–1308. DOI: [10.1175/2009WAF222247.1](https://doi.org/10.1175/2009WAF222247.1).
- Kendall, A. and Y. Gal (2017). “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” In: *Advances in Neural Information Processing Systems 30*, pp. 5574–5584. URL: <https://proceedings.neurips.cc/paper/2017/hash/2650d6089a6d640c5e85b2b88265dc2b-Abstract.html>.
- Kettunen, M., E. Härkönen, and J. Lehtinen (2019). *E-LPIPS: Robust Perceptual Image Similarity via Random Transformation Ensembles*. arXiv: [1906.03973](https://arxiv.org/abs/1906.03973) [cs].
- Kiefer, J. and J. Wolfowitz (1952). “Stochastic Estimation of the Maximum of a Regression Function”. In: *The Annals of Mathematical Statistics* 23.3, pp. 462–466. DOI: [10.1214/aoms/1177729392](https://doi.org/10.1214/aoms/1177729392).
- Kim, B., V. C. Azevedo, M. Gross, and B. Solenthaler (2019a). “Transport-Based Neural Style Transfer for Smoke Simulations”. In: *ACM Transactions on Graphics* 38.6, pp. 1–11. ISSN: 0730-0301, 1557-7368. DOI: [10.1145/3355089.3356560](https://doi.org/10.1145/3355089.3356560).
- Kim, B., V. C. Azevedo, M. Gross, and B. Solenthaler (2020). “Lagrangian Neural Style Transfer for Fluids”. In: *ACM Transactions on Graphics* 39.4. ISSN: 0730-0301, 1557-7368. DOI: [10.1145/3386569.3392473](https://doi.org/10.1145/3386569.3392473).
- Kim, B., V. C. Azevedo, N. Thuerey, T. Kim, M. H. Gross, and B. Solenthaler (2019b). “Deep Fluids: A Generative Network for Parameterized Fluid Simulations”. In: *Computer Graphics Forum* 38.2, pp. 59–70. DOI: [10.1111/cgf.13619](https://doi.org/10.1111/cgf.13619).
- Kim, J. and S. Lee (2017). “Deep Learning of Human Visual Sensitivity in Image Quality Assessment Framework”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1969–1977. DOI: [10.1109/CVPR.2017.213](https://doi.org/10.1109/CVPR.2017.213).
- Kim, T., N. Thuerey, D. James, and M. Gross (2008). “Wavelet Turbulence for Fluid Simulation”. In: *ACM Transactions on Graphics* 27.3, pp. 1–6. ISSN: 0730-0301. DOI: [10.1145/1360612.1360649](https://doi.org/10.1145/1360612.1360649).
- Kingma, D. P. and J. Ba (2015). “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations (ICLR 2015)*. URL: <http://arxiv.org/abs/1412.6980>.
- Kingma, D. P., T. Salimans, and M. Welling (2015). “Variational Dropout and the Local Reparameterization Trick”. In: *Advances in Neural Information Processing Systems 28*, pp. 2575–2583. URL: <https://proceedings.neurips.cc/paper/2015/hash/bc7316929fe1545bf0b98d114ee3ecb8-Abstract.html>.
- Kingma, D. P. and M. Welling (2014). “Auto-Encoding Variational Bayes”. In: *2nd International Conference on Learning Representations (ICLR 2014)*. URL: <http://arxiv.org/abs/1312.6114>.
- Koch, G., R. Zemel, R. Salakhutdinov, et al. (2015). “Siamese Neural Networks for One-Shot Image Recognition”. In: *ICML Deep Learning Workshop*. Vol. 2. URL: <https://cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf>.

- Kochkov, D., J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and Stephan Hoyer (2021). “Machine Learning–Accelerated Computational Fluid Dynamics”. In: *Proceedings of the National Academy of Sciences* 118.21, e2101784118. DOI: [10.1073/pnas.2101784118](https://doi.org/10.1073/pnas.2101784118).
- Kohl, G. (2019). “Accuracy Evaluation of Numerical Simulation Methods with CNNs”. MA thesis. Technical University of Munich.
- Kohl, G., L.-W. Chen, and N. Thuerey (2023). “Learning Similarity Metrics for Volumetric Simulations with Multiscale CNNs”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37–7, pp. 8351–8359. DOI: [10.1609/aaai.v37i7.26007](https://doi.org/10.1609/aaai.v37i7.26007).
- Kohl, G., L.-W. Chen, and N. Thuerey (2024). *Benchmarking Autoregressive Conditional Diffusion Models for Turbulent Flow Simulation*. arXiv: [2309.01745](https://arxiv.org/abs/2309.01745) [cs, physics].
- Kohl, G., K. Um, and N. Thuerey (2020). “Learning Similarity Metrics for Numerical Simulations”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*. Vol. 119, pp. 5349–5360. URL: <https://proceedings.mlr.press/v119/kohl20a.html>.
- Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems* 25, pp. 1106–1114. URL: <https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html>.
- Kullback, S. and R. A. Leibler (1951). “On Information and Sufficiency”. In: *The annals of mathematical statistics* 22.1, pp. 79–86. DOI: [10.1214/aoms/1177729694](https://doi.org/10.1214/aoms/1177729694).
- Kumar, M., N. Houlsby, N. Kalchbrenner, and E. D. Cubuk (2022). “Do Better ImageNet Classifiers Assess Perceptual Similarity Better?” In: *Transactions on Machine Learning Research* 2022. URL: <https://openreview.net/forum?id=qrGKGZZvH0>.
- Kuramoto, Y. (1978). “Diffusion-Induced Chaos in Reaction Systems”. In: *Progress of Theoretical Physics Supplement* 64, pp. 346–367. ISSN: 0375-9687. DOI: [10.1143/PTPS.64.346](https://doi.org/10.1143/PTPS.64.346).
- Ladicky, L., S. Jeong, B. Solenthaler, M. Pollefeys, and M. H. Gross (2015). “Data-Driven Fluid Simulations Using Regression Forests”. In: *ACM Transactions on Graphics* 34.6, 199:1–199:9. DOI: [10.1145/2816795.2818129](https://doi.org/10.1145/2816795.2818129).
- Lao, S., Y. Gong, S. Shi, S. Yang, T. Wu, J. Wang, W. Xia, and Y. Yang (2022). “Attentions Help CNNs See Better: Attention-Based Hybrid Image Quality Assessment Network”. In: *2022 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 1139–1148. DOI: [10.1109/CVPRW56347.2022.00123](https://doi.org/10.1109/CVPRW56347.2022.00123).
- Larson, E. C. and D. M. Chandler (2010). “Most Apparent Distortion: Full-Reference Image Quality Assessment and the Role of Strategy”. In: *Journal of Electronic Imaging* 19.1. DOI: [10.1117/1.3267105](https://doi.org/10.1117/1.3267105).
- LeCun, Y., B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel (1989). “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Computation* 1.4, pp. 541–551. DOI: [10.1162/NECO.1989.1.4.541](https://doi.org/10.1162/NECO.1989.1.4.541).
- LeVeque, R. J. (2007). *Finite Difference Methods for Ordinary and Partial Differential Equations - Steady-State and Time-Dependent Problems*. SIAM. ISBN: 978-0-89871-629-0. DOI: [10.1137/1.9780898717839](https://doi.org/10.1137/1.9780898717839).
- Li, Z., N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar (2021). “Fourier Neural Operator for Parametric Partial Differential Equations”.

- In: *9th International Conference on Learning Representations (ICLR 2021)*. URL: <https://openreview.net/forum?id=c8P9NQVtmm0>.
- Lienen, M., J. Hansen-Palmus, D. Lüdke, and S. Günnemann (2023). *From Zero to Turbulence: Generative Modeling for 3D Flow Simulation*. arXiv: [2306.01776](https://arxiv.org/abs/2306.01776) [physics].
- Lin, Z., T. S. Hahm, W. Lee, W. M. Tang, and R. B. White (1998). “Turbulent Transport Reduction by Zonal Flows: Massively Parallel Simulations”. In: *Science* 281.5384, pp. 1835–1837. DOI: [10.1126/science.281.5384.1835](https://doi.org/10.1126/science.281.5384.1835).
- Lippe, P., B. Veeling, P. Perdikaris, R. E. Turner, and J. Brandstetter (2023). “PDE-Refiner: Achieving Accurate Long Rollouts with Neural PDE Solvers”. In: *Advances in Neural Information Processing Systems 36*. URL: http://papers.nips.cc/paper_files/paper/2023/hash/d529b943af3dba734f8a7d49efcb6d09-Abstract-Conference.html.
- List, B., L.-W. Chen, and N. Thuerey (2022). “Learned Turbulence Modelling with Differentiable Fluid Solvers: Physics-Based Loss Functions and Optimisation Horizons”. In: *Journal of Fluid Mechanics* 949, A25. DOI: [10.1017/jfm.2022.738](https://doi.org/10.1017/jfm.2022.738).
- Liu, X., M. Pedersen, and J. Y. Hardeberg (2014). “CID:IQ - A New Image Quality Database”. In: *Image and Signal Processing (ICISP)*. Vol. 8509, pp. 193–202. DOI: [10.1007/978-3-319-07998-1_22](https://doi.org/10.1007/978-3-319-07998-1_22).
- Madhusudana, P. C., N. Birkbeck, Y. Wang, B. Adsumilli, and A. C. Bovik (2022). “Image Quality Assessment Using Contrastive Learning”. In: *IEEE Transactions on Image Processing* 31, pp. 4149–4161. DOI: [10.1109/TIP.2022.3181496](https://doi.org/10.1109/TIP.2022.3181496).
- Meilă, M. (2007). “Comparing Clusterings—an Information Based Distance”. In: *Journal of Multivariate Analysis* 98.5, pp. 873–895. ISSN: 0047-259X. DOI: [10.1016/j.jmva.2006.11.013](https://doi.org/10.1016/j.jmva.2006.11.013).
- Mialon, G., Q. Garrido, H. Lawrence, D. Rehman, Y. LeCun, and B. T. Kiani (2023). “Self-Supervised Learning with Lie Symmetries for Partial Differential Equations”. In: *Advances in Neural Information Processing Systems 36*. URL: http://papers.nips.cc/paper_files/paper/2023/hash/5c46ae130105fa012da0446126c01d1d-Abstract-Conference.html.
- Mier, J. C., E. Huang, H. Talebi, F. Yang, and P. Milanfar (2021). “Deep Perceptual Image Quality Assessment for Compression”. In: *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 1484–1488. DOI: [10.1109/ICIP42928.2021.9506217](https://doi.org/10.1109/ICIP42928.2021.9506217).
- Moin, P. and K. Mahesh (1998). “Direct Numerical Simulation: A Tool in Turbulence Research”. In: *Annual Review of Fluid Mechanics* 30.1, pp. 539–578. DOI: [10.1146/annurev.fluid.30.1.539](https://doi.org/10.1146/annurev.fluid.30.1.539).
- Monaghan, J. J. (1992). “Smoothed Particle Hydrodynamics”. In: *Annual Review of Astronomy and Astrophysics* 30.1, pp. 543–574. DOI: [10.1146/annurev.aa.30.090192.002551](https://doi.org/10.1146/annurev.aa.30.090192.002551).
- Nilsson, J. and T. Akenine-Möller (2020). *Understanding SSIM*. arXiv: [2006.13846](https://arxiv.org/abs/2006.13846) [cs, eess].
- Oberkampf, W. L., T. G. Trucano, and C. Hirsch (2004). “Verification, Validation, and Predictive Capability in Computational Engineering and Physics”. In: *Applied Mechanics Reviews* 57.5, pp. 345–384. DOI: [10.1115/1.1767847](https://doi.org/10.1115/1.1767847).
- Olufsen, M. S., C. S. Peskin, W. Y. Kim, E. M. Pedersen, A. Nadim, and J. Larsen (2000). “Numerical Simulation and Experimental Validation of Blood Flow in Arteries with Structured-

- Tree Outflow Conditions”. In: *Annals of Biomedical Engineering* 28.11, pp. 1281–1299. DOI: [10.1114/1.1326031](https://doi.org/10.1114/1.1326031).
- Pearson, K. (1920). “Notes on the History of Correlation”. In: *Biometrika* 13.1, pp. 25–45. DOI: [10.1093/biomet/13.1.25](https://doi.org/10.1093/biomet/13.1.25).
- Perlman, E., R. Burns, Y. Li, and C. Meneveau (2007). “Data Exploration of Turbulence Simulations Using a Database Cluster”. In: *Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing*, pp. 1–11. DOI: [10.1145/1362622.1362654](https://doi.org/10.1145/1362622.1362654).
- Pfaff, T., M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia (2021). “Learning Mesh-Based Simulation with Graph Networks”. In: *9th International Conference on Learning Representations (ICLR 2021)*. URL: https://openreview.net/forum?id=roNqYL0_XP.
- Pitsch, H. (2006). “Large-Eddy Simulation of Turbulent Combustion”. In: *Annual Review of Fluid Mechanics* 38, pp. 453–482. DOI: [10.1146/annurev.fluid.38.050304.092133](https://doi.org/10.1146/annurev.fluid.38.050304.092133).
- Ponomarenko, N., L. Jin, O. Ieremeiev, V. Lukin, K. Egiazarian, J. Astola, B. Vozel, K. Chehdi, M. Carli, F. Battisti, and C. C. J. Kuo (2015). “Image Database TID2013: Peculiarities, Results and Perspectives”. In: *Signal Processing-Image Communication* 30, pp. 57–77. DOI: [10.1016/j.image.2014.10.009](https://doi.org/10.1016/j.image.2014.10.009).
- Pope, S. (2000). *Turbulent Flows*. Cambridge University Press. ISBN: 978-0-511-84053-1. DOI: [10.1017/CB09780511840531](https://doi.org/10.1017/CB09780511840531).
- Prashnani, E., H. Cai, Y. Mostofi, and P. Sen (2018). “PieAPP: Perceptual Image-Error Assessment through Pairwise Preference”. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1808–1817. DOI: [10.1109/CVPR.2018.00194](https://doi.org/10.1109/CVPR.2018.00194).
- Prince, S. (2023). *Understanding Deep Learning*. MIT Press. ISBN: 978-0-262-04864-4.
- Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever (2019). “Language Models Are Unsupervised Multitask Learners”. In: *OpenAI*. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
- Raissi, M., P. Perdikaris, and G. E. Karniadakis (2019). “Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations”. In: *Journal of Computational Physics* 378, pp. 686–707. DOI: [10.1016/J.JCP.2018.10.045](https://doi.org/10.1016/J.JCP.2018.10.045).
- Ramachandra, B., M. Jones, and R. R. Vatsavai (2021). “Perceptual Metric Learning for Video Anomaly Detection”. In: *Machine Vision and Applications* 32.3, p. 63. DOI: [10.1007/S00138-021-01187-5](https://doi.org/10.1007/S00138-021-01187-5).
- Ramesh, A., M. Goyal, G. Goh, and S. Gray (2021). “DALL·E: Creating Images from Text”. In: *OpenAI*. URL: <https://openai.com/research/dall-e>.
- Rasp, S., P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, and N. Thuerey (2020). “Weather-Bench: A Benchmark Dataset for Data-Driven Weather Forecasting”. In: *Journal of Advances in Modeling Earth Systems* 12.11, e2020MS002203. DOI: [10.1029/2020MS002203](https://doi.org/10.1029/2020MS002203).
- Reddy, J. N. (2019). *Introduction to the Finite Element Method*. McGraw-Hill Education. ISBN: 1-259-86190-2.
- Ronneberger, O., P. Fischer, and T. Brox (2015). “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351, pp. 234–241. DOI: [10.1007/978-3-319-24574-4_28](https://doi.org/10.1007/978-3-319-24574-4_28).

- Ruder, M., A. Dosovitskiy, and T. Brox (2016). “Artistic Style Transfer for Videos”. In: *Pattern Recognition (GCPR)*, pp. 26–36. DOI: [10.1007/978-3-319-45886-1_3](https://doi.org/10.1007/978-3-319-45886-1_3).
- Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). “Learning Representations by Back-Propagating Errors”. In: *Nature* 323.6088, pp. 533–536. ISSN: 1476-4687. DOI: [10.1038/323533a0](https://doi.org/10.1038/323533a0).
- Saharia, C., W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, S. K. S. Ghasemipour, R. G. Lopes, B. K. Ayan, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi (2022). “Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding”. In: *Advances in Neural Information Processing Systems 35*. URL: http://papers.nips.cc/paper_files/paper/2022/hash/ec795aeadae0b7d230fa35cbaf04c041-Abstract-Conference.html.
- Salimans, T., I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen (2016). “Improved Techniques for Training GANs”. In: *Advances in Neural Information Processing Systems 29*, pp. 2226–2234. URL: <https://proceedings.neurips.cc/paper/2016/hash/8a3363abe792db2d8761d6403605aeb7-Abstract.html>.
- Sanchez-Gonzalez, A., J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia (2020). “Learning to Simulate Complex Physics with Graph Networks”. In: *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*. Vol. 119, pp. 8459–8468. URL: <https://proceedings.mlr.press/v119/sanchez-gonzalez20a.html>.
- Scarselli, F., M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini (2009). “The Graph Neural Network Model”. In: *IEEE Transactions on Neural Networks* 20.1, pp. 61–80. DOI: [10.1109/TNN.2008.2005605](https://doi.org/10.1109/TNN.2008.2005605).
- Shu, D., Z. Li, and A. B. Farimani (2023). “A Physics-Informed Diffusion Model for High-Fidelity Flow Field Reconstruction”. In: *Journal of Computational Physics* 478, p. 111972. DOI: [10.1016/j.jcp.2023.111972](https://doi.org/10.1016/j.jcp.2023.111972).
- Simcenter STAR-CCM+* (2004). Siemens Digital Industries Software. URL: <https://plm.sw.siemens.com/en-US/simcenter/fluids-thermal-simulation/star-ccm>.
- Simonyan, K., A. Vedaldi, and A. Zisserman (2014). “Deep inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *2nd International Conference on Learning Representations (ICLR 2014)*. URL: <http://arxiv.org/abs/1312.6034>.
- Simonyan, K. and A. Zisserman (2015). “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations (ICLR 2015)*. URL: <http://arxiv.org/abs/1409.1556>.
- Sirignano, J. A., J. F. MacArt, and J. B. Freund (2020). “DPM: A Deep Learning PDE Augmentation Method with Application to Large-Eddy Simulation”. In: *Journal of Computational Physics* 423, p. 109811. DOI: [10.1016/j.jcp.2020.109811](https://doi.org/10.1016/j.jcp.2020.109811).
- Sivashinsky, G. (1977). “Nonlinear Analysis of Hydrodynamic Instability in Laminar Flames—I. Derivation of Basic Equations”. In: *Acta Astronautica* 4.11, pp. 1177–1206. ISSN: 0094-5765. DOI: [10.1016/0094-5765\(77\)90096-0](https://doi.org/10.1016/0094-5765(77)90096-0).
- Song, Y., J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole (2021). “Score-Based Generative Modeling through Stochastic Differential Equations”. In: *9th International Conference on Learning Representations (ICLR 2021)*. URL: <https://openreview.net/forum?id=PXTIG12RRHS>.

- Spearman, C. (1904). “The Proof and Measurement of Association between Two Things”. In: *The American Journal of Psychology* 15.1, pp. 72–101. DOI: [10.2307/1412159](https://doi.org/10.2307/1412159).
- Srivastava, N., G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.1, pp. 1929–1958. URL: <http://jmlr.org/papers/v15/srivastava14a.html>.
- Stachenfeld, K. L., D. B. Fielding, D. Kochkov, M. D. Cranmer, T. Pfaff, J. Godwin, C. Cui, S. Ho, P. W. Battaglia, and A. Sanchez-Gonzalez (2022). “Learned Coarse Models for Efficient Turbulence Simulation”. In: *10th International Conference on Learning Representations (ICLR 2022)*. URL: <https://openreview.net/forum?id=msRBojTz-Nh>.
- Stam, J. (1999). “Stable Fluids”. In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 121–128. DOI: [10.1145/311535.311548](https://doi.org/10.1145/311535.311548).
- Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015). “Going Deeper with Convolutions”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9. DOI: [10.1109/CVPR.2015.7298594](https://doi.org/10.1109/CVPR.2015.7298594).
- Takamoto, M., F. Alesiani, and M. Niepert (2023). “Learning Neural PDE Solvers with Parameter-Guided Channel Attention”. In: *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*. Vol. 202, pp. 33448–33467. URL: <https://proceedings.mlr.press/v202/takamoto23a.html>.
- Takamoto, M., T. Praditia, R. Leiteritz, D. MacKinlay, F. Alesiani, D. Pflüger, and M. Niepert (2022). “PDEBench: An Extensive Benchmark for Scientific Machine Learning”. In: *Advances in Neural Information Processing Systems 35*. URL: http://papers.nips.cc/paper_files/paper/2022/hash/0a9747136d411fb83f0cf81820d44afb-Abstract-Datasets_and_Benchmarks.html.
- Talebi, H. and P. Milanfar (2018a). “Learned Perceptual Image Enhancement”. In: *2018 IEEE International Conference on Computational Photography (ICCP)*. DOI: [10.1109/ICCPHOT.2018.8368474](https://doi.org/10.1109/ICCPHOT.2018.8368474).
- Talebi, H. and P. Milanfar (2018b). “NIMA: Neural Image Assessment”. In: *IEEE Transactions on Image Processing* 27.8, pp. 3998–4011. DOI: [10.1109/TIP.2018.2831899](https://doi.org/10.1109/TIP.2018.2831899).
- Tan, M. and Q. V. Le (2019). “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*. Vol. 97, pp. 6105–6114. URL: <http://proceedings.mlr.press/v97/tan19a.html>.
- Thuerey, N., P. Holl, M. Mueller, P. Schnell, F. Trost, and K. Um (2022). *Physics-Based Deep Learning*. arXiv: [2109.05237 \[physics\]](https://arxiv.org/abs/2109.05237).
- Thuerey, N., K. Weissenow, L. Prantl, and X. Hu (2020). “Deep Learning Methods for Reynolds-Averaged Navier-Stokes Simulations of Airfoil Flows”. In: *AIAA Journal* 58.1, pp. 25–36. DOI: [10.2514/1.J058291](https://doi.org/10.2514/1.J058291).
- Tompson, J., K. Schlachter, P. Sprechmann, and K. Perlin (2017). “Accelerating Eulerian Fluid Simulation with Convolutional Networks”. In: *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*. Vol. 70. Proceedings of Machine Learning Research, pp. 3424–3433. URL: <https://proceedings.mlr.press/v70/tompson17a.html>.

- Touvron, H., T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample (2023). *LLaMA: Open and Efficient Foundation Language Models*. arXiv: [2302.13971](https://arxiv.org/abs/2302.13971) [cs].
- Tsubota, K., H. Akutsu, and K. Aizawa (2022). “Evaluating the Stability of Deep Image Quality Assessment with Respect to Image Scaling”. In: *IEICE Transactions on Information Systems* 105-D.10, pp. 1829–1833. DOI: [10.1587/TRANSINF.2022EDL8025](https://doi.org/10.1587/TRANSINF.2022EDL8025).
- Um, K., R. Brand, Y. Fei, P. Holl, and N. Thuerey (2020). “Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers”. In: *Advances in Neural Information Processing Systems 33*. URL: <https://proceedings.neurips.cc/paper/2020/hash/43e4e6a6f341e00671e123714de019a8-Abstract.html>.
- Um, K., X. Hu, and N. Thuerey (2017). “Perceptual Evaluation of Liquid Simulation Methods”. In: *ACM Transactions on Graphics* 36.4. DOI: [10.1145/3072959.3073633](https://doi.org/10.1145/3072959.3073633).
- Um, K., X. Hu, B. Wang, and N. Thuerey (2021). “Spot the Difference: Accuracy of Numerical Simulations via the Human Visual System”. In: *ACM Transactions on Applied Perception* 18.2, 6:1–6:15. DOI: [10.1145/3449064](https://doi.org/10.1145/3449064).
- Ummenhofer, B., L. Prantl, N. Thuerey, and V. Koltun (2020). “Lagrangian Fluid Simulation with Continuous Convolutions”. In: *8th International Conference on Learning Representations (ICLR 2020)*. URL: <https://openreview.net/forum?id=B1lDoJSYDH>.
- Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). “Attention Is All You Need”. In: *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. URL: <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>.
- Versteeg, H. K. and W. Malalasekera (1995). *An Introduction to Computational Fluid Dynamics - the Finite Volume Method*. Addison-Wesley-Longman. ISBN: 978-0-582-21884-0.
- Wang, R., K. Kashinath, M. Mustafa, A. Albert, and R. Yu (2020). “Towards Physics-Informed Deep Learning for Turbulent Flow Prediction”. In: *26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1457–1466. DOI: [10.1145/3394486.3403198](https://doi.org/10.1145/3394486.3403198).
- Wang, R., R. Walters, and R. Yu (2021). “Incorporating Symmetry into Deep Dynamics Models for Improved Generalization”. In: *9th International Conference on Learning Representations (ICLR 2021)*. URL: <https://arxiv.org/abs/2002.03061>.
- Wang, X. and A. Gupta (2015). “Unsupervised Learning of Visual Representations Using Videos”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2794–2802. DOI: [10.1109/ICCV.2015.320](https://doi.org/10.1109/ICCV.2015.320).
- Wang, Z., A. C. Bovik, H. R. Sheikh, and E. Simoncelli (2004). “Image Quality Assessment: From Error Visibility to Structural Similarity”. In: *IEEE Transactions on Image Processing* 13.4, pp. 600–612. DOI: [10.1109/TIP.2003.819861](https://doi.org/10.1109/TIP.2003.819861).
- Wang, Z., E. P. Simoncelli, and A. C. Bovik (2003). “Multiscale Structural Similarity for Image Quality Assessment”. In: *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers (2003)*. Vol. 2, pp. 1398–1402. DOI: [10.1109/ACSSC.2003.1292216](https://doi.org/10.1109/ACSSC.2003.1292216).
- Weiler, M., M. Geiger, M. Welling, W. Boomsma, and T. Cohen (2018). “3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data”. In: *Advances in Neural Information Processing Systems 31*, pp. 10402–10413. URL: <https://proceedings.neurips.cc/paper/2018/hash/488e4104520c6aab692863cc1dba45af-Abstract.html>.

- Weller, H. G., G. Tabor, H. Jasak, and C. Fureby (1998). “A Tensorial Approach to Computational Continuum Mechanics Using Object-Oriented Techniques”. In: *Computer in Physics* 12.6, pp. 620–631. ISSN: 0894-1866. DOI: [10.1063/1.168744](https://doi.org/10.1063/1.168744).
- Wiewel, S., M. Becher, and N. Thuerey (2019). “Latent Space Physics: Towards Learning the Temporal Evolution of Fluid Flow”. In: *Computer Graphics Forum* 38.2, pp. 71–82. ISSN: 1467-8659. DOI: [10.1111/cgf.13620](https://doi.org/10.1111/cgf.13620).
- Wiewel, S., B. Kim, V. C. Azevedo, B. Solenthaler, and N. Thuerey (2020). “Latent Space Subdivision: Stable and Controllable Time Predictions for Fluid Flow”. In: *Computer Graphics Forum* 39.8, pp. 15–25. ISSN: 1467-8659. DOI: [10.1111/cgf.14097](https://doi.org/10.1111/cgf.14097).
- Xie, Y., E. Franz, M. Chu, and N. Thuerey (2018). “tempoGAN: A Temporally Coherent, Volumetric GAN for Super-Resolution Fluid Flow”. In: *ACM Transactions on Graphics* 37.4, p. 95. DOI: [10.1145/3197517.3201304](https://doi.org/10.1145/3197517.3201304).
- Zeiler, M. D. and R. Fergus (2014). “Visualizing and Understanding Convolutional Networks”. In: *Computer Vision - ECCV 2014*. Vol. 8689, pp. 818–833. DOI: [10.1007/978-3-319-10590-1_53](https://doi.org/10.1007/978-3-319-10590-1_53).
- Zhang, R., P. Isola, A. A. Efros, E. Shechtman, and O. Wang (2018). “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric”. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595. DOI: [10.1109/CVPR.2018.00068](https://doi.org/10.1109/CVPR.2018.00068).
- Zhang, Y. and Z. Duan (2017). “IMINET: Convolutional Semi-Siamese Networks for Sound Search by Vocal Imitation”. In: *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 304–308. DOI: [10.1109/TASLP.2018.2868428](https://doi.org/10.1109/TASLP.2018.2868428).
- Zhu, Y. and R. Bridson (2005). “Animating Sand As a Fluid”. In: *ACM Transactions on Graphics*, pp. 965–972. DOI: [10.1145/1186822.1073298](https://doi.org/10.1145/1186822.1073298).

Learning Similarity Metrics for Numerical Simulations

Georg Kohl¹ Kiwon Um¹ Nils Thuerey¹

Abstract

We propose a neural network-based approach that computes a stable and generalizing metric (*LSiM*) to compare data from a variety of numerical simulation sources. We focus on scalar time-dependent 2D data that commonly arises from motion and transport-based partial differential equations (PDEs). Our method employs a Siamese network architecture that is motivated by the mathematical properties of a metric. We leverage a controllable data generation setup with PDE solvers to create increasingly different outputs from a reference simulation in a controlled environment. A central component of our learned metric is a specialized loss function that introduces knowledge about the correlation between single data samples into the training process. To demonstrate that the proposed approach outperforms existing metrics for vector spaces and other learned, image-based metrics, we evaluate the different methods on a large range of test data. Additionally, we analyze generalization benefits of an adjustable training data difficulty and demonstrate the robustness of *LSiM* via an evaluation on three real-world data sets.

1. Introduction

Evaluating computational tasks for complex data sets is a fundamental problem in all computational disciplines. Regular vector space metrics, such as the L^2 distance, were shown to be very unreliable (Wang et al., 2004; Zhang et al., 2018), and the advent of deep learning techniques with convolutional neural networks (CNNs) made it possible to more reliably evaluate complex data domains such as natural images, texts (Benajiba et al., 2018), or speech (Wang et al., 2018). Our central aim is to demonstrate the usefulness of

¹Department of Informatics, Technical University of Munich, Munich, Germany. Correspondence to: Georg Kohl <georg.kohl@tum.de>.

CNN-based evaluations in the context of numerical simulations. These simulations are the basis for a wide range of applications ranging from blood flow simulations to aircraft design. Specifically, we propose a novel learned simulation metric (*LSiM*) that allows for a reliable similarity evaluation of simulation data.

Potential applications of such a metric arise in all areas where numerical simulations are performed or similar data is gathered from observations. For example, accurate evaluations of existing and new simulation methods with respect to a known ground truth solution (Oberkampf et al., 2004) can be performed more reliably than with a regular vector norm. Another good example is weather data for which complex transport processes and chemical reactions make in-place comparisons with common metrics unreliable (Jolliffe & Stephenson, 2012). Likewise, the long-standing, open questions of turbulence (Moin & Mahesh, 1998; Lin et al., 1998) can benefit from improved methods for measuring the similarity and differences in data sets and observations.

In this work, we focus on field data, i.e., dense grids of scalar values, similar to images, which were generated with known partial differential equations (PDEs) in order to ensure the availability of ground truth solutions. While we focus on 2D data in the following to make comparisons with existing techniques from imaging applications possible, our approach naturally extends to higher dimensions. Every sample of this 2D data can be regarded a high dimensional vector, so metrics on the corresponding vector space are applicable to evaluate similarities. These metrics, in the following denoted as *shallow metrics*, are typically simple, element-wise functions such as L^1 or L^2 distances. Their inherent problem is that they cannot compare structures on different scales or contextual information.

Many practical problems require solutions over time and need a vast number of non-linear operations that often result in substantial changes of the solutions even for small changes of the inputs. Hence, despite being based on known, continuous formulations, these systems can be seen as *chaotic*. We illustrate this behavior in Fig. 1, where two smoke flows are compared to a reference simulation. A single simulation parameter was varied for these examples, and a visual inspection shows that smoke plume (a) is more similar to the reference. This matches the data generation

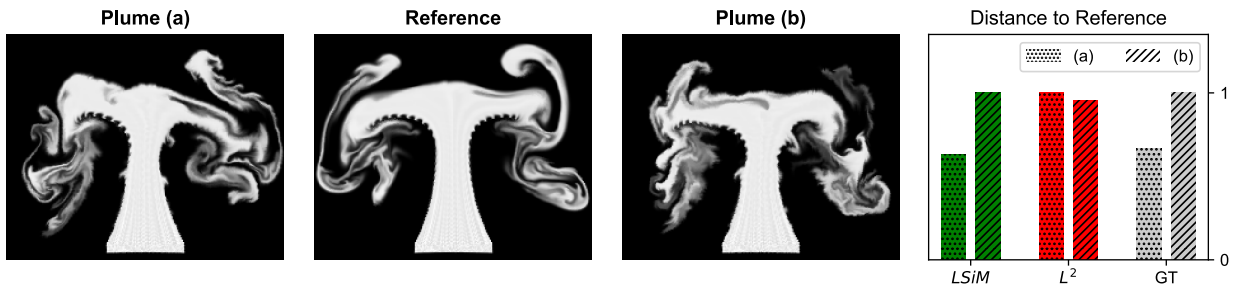


Figure 1. Example of field data from a fluid simulation of hot smoke with normalized distances for different metrics. Our method (*LSiM*, green) approximates the ground truth distances (GT, gray) determined by the data generation method best, i.e., version (a) is closer to the ground truth data than (b). An L^2 metric (red) erroneously yields a reversed ordering.

process: version (a) has a significantly smaller parameter change than (b) as shown in the inset graph on the right. *LSiM* robustly predicts the ground truth distances while the L^2 metric labels plume (b) as more similar. In our work, we focus on retrieving the relative distances of simulated data sets. Thus, we do not aim for retrieving the absolute parameter change but a relative distance that preserves ordering with respect to this parameter.

Using existing image metrics based on CNNs for this problem is not optimal either: natural images only cover a small fraction of the space of possible 2D data, and numerical simulation outputs are located in a fundamentally different data manifold within this space. Hence, there are crucial aspects that cannot be captured by purely learning from photographs. Furthermore, we have full control over the data generation process for simulation data. As a result, we can create arbitrary amounts of training data with gradual changes and a ground truth ordering. With this data, we can learn a metric that is not only able to directly extract and use features but also encodes interactions between them. The central contributions of our work are as follows:

- We propose a Siamese network architecture with feature map normalization, which is able to learn a metric that generalizes well to unseen motion and transport-based simulation methods.
- We propose a novel loss function that combines a correlation loss term with a mean squared error to improve the accuracy of the learned metric.
- In addition, we show how a data generation approach for numerical simulations can be employed to train networks with general and robust feature extractors for metric calculations.

Our source code, data sets, and final model are available at <https://github.com/tum-pbs/LSiM>.

2. Related Work

One of the earliest methods to go beyond using simple metrics based on L^p norms for natural images was the structural

similarity index (Wang et al., 2004). Despite improvements, this method can still be considered a shallow metric. Over the years, multiple large databases for human evaluations of natural images were presented, for instance, CSIQ (Larson & Chandler, 2010), TID2013 (Ponomarenko et al., 2015), and CID:IQ (Liu et al., 2014). With this data and the discovery that CNNs can create very powerful feature extractors that are able to recognize patterns and structures, deep feature maps quickly became established as means for evaluation (Amirshahi et al., 2016; Berardino et al., 2017; Bosse et al., 2016; Kang et al., 2014; Kim & Lee, 2017). Recently, these methods were improved by predicting the distribution of human evaluations instead of directly learning distance values (Prashnani et al., 2018; Talebi & Milanfar, 2018b). Zhang et al. compared different architecture and levels of supervision, and showed that metrics can be interpreted as a transfer learning approach by applying a linear weighting to the feature maps of any network architecture to form the image metric *LPIPS v0.1*. Typical use cases of these image-based CNN metrics are computer vision tasks such as detail enhancement (Talebi & Milanfar, 2018a), style transfer, and super-resolution (Johnson et al., 2016). Generative adversarial networks also leverage CNN-based losses by training a discriminator network in parallel to the generation task (Dosovitskiy & Brox, 2016).

Siamese network architectures are known to work well for a variety of comparison tasks such as audio (Zhang & Duan, 2017), satellite images (He et al., 2019), or the similarity of interior product designs (Bell & Bala, 2015). Furthermore, they yield robust object trackers (Bertinetto et al., 2016), algorithms for image patch matching (Hanif, 2019), and for descriptors for fluid flow synthesis (Chu & Thuerey, 2017). Inspired by these studies, we use a similar Siamese neural network architecture for our metric learning task. In contrast to other work on self-supervised learning that utilizes spatial or temporal changes to learn meaningful representations (Agrawal et al., 2015; Wang & Gupta, 2015), our method does not rely on tracked keypoints in the data.

While correlation terms have been used for learning joint representations by maximizing correlation of projected

views (Chandar et al., 2016) and are popular for style transfer applications via the Gram matrix (Ruder et al., 2016), they were not used for learning distance metrics. As we demonstrate below, they can yield significant improvements in terms of the inferred distances.

Similarity metrics for numerical simulations are a topic of ongoing investigation. A variety of specialized metrics have been proposed to overcome the limitations of L^p norms, such as the displacement and amplitude score from the area of weather forecasting (Keil & Craig, 2009) as well as permutation based metrics for energy consumption forecasting (Haben et al., 2014). Turbulent flows, on the other hand, are often evaluated in terms of aggregated frequency spectra (Pitsch, 2006). Crowd-sourced evaluations based on the human visual system were also proposed to evaluate simulation methods for physics-based animation (Um et al., 2017) and for comparing non-oscillatory discretization schemes (Um et al., 2019). These results indicate that visual evaluations in the context of field data are possible and robust, but they require extensive (and potentially expensive) user studies. Additionally, our method naturally extends to higher dimensions, while human evaluations inherently rely on projections with at most two spatial and one time dimension.

3. Constructing a CNN-based Metric

In the following, we explain our considerations when employing CNNs as evaluation metrics. For a comparison that corresponds to our intuitive understanding of distances, an underlying *metric* has to obey certain criteria. More precisely, a function $m : \mathbb{I} \times \mathbb{I} \rightarrow [0, \infty)$ is a metric on its input space \mathbb{I} if it satisfies the following properties $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{I}$:

$$m(\mathbf{x}, \mathbf{y}) \geq 0 \quad \text{non-negativity} \quad (1)$$

$$m(\mathbf{x}, \mathbf{y}) = m(\mathbf{y}, \mathbf{x}) \quad \text{symmetry} \quad (2)$$

$$m(\mathbf{x}, \mathbf{y}) \leq m(\mathbf{x}, \mathbf{z}) + m(\mathbf{z}, \mathbf{y}) \quad \text{triangle ineq.} \quad (3)$$

$$m(\mathbf{x}, \mathbf{y}) = 0 \iff \mathbf{x} = \mathbf{y} \quad \text{identity of indisc.} \quad (4)$$

The properties (1) and (2) are crucial as distances should be symmetric and have a clear lower bound. Eq. (3) ensures

that direct distances cannot be longer than a detour. Property (4), on the other hand, is not really useful for discrete operations as approximation errors and floating point operations can easily lead to a distance of zero for slightly different inputs. Hence, we focus on a relaxed, more meaningful definition $m(\mathbf{x}, \mathbf{x}) = 0 \forall \mathbf{x} \in \mathbb{I}$, which leads to a so-called *pseudometric*. It allows for a distance of zero for different inputs but has to be able to spot identical inputs.

We realize these requirements for a pseudometric with an architecture that follows popular perceptual metrics such as *LPIPS*: The activations of a CNN are compared in latent space, accumulated with a set of weights, and the resulting per-feature distances are aggregated to produce a final distance value. Fig. 2 gives a visual overview of this process.

To show that the proposed Siamese architecture by construction qualifies as a pseudometric, the function

$$m(\mathbf{x}, \mathbf{y}) = m_2(m_1(\mathbf{x}), m_1(\mathbf{y}))$$

computed by our network is split into two parts: $m_1 : \mathbb{I} \rightarrow \mathbb{L}$ to compute the latent space embeddings $\tilde{\mathbf{x}} = m_1(\mathbf{x})$, $\tilde{\mathbf{y}} = m_1(\mathbf{y})$ from each input, and $m_2 : \mathbb{L} \rightarrow [0, \infty)$ to compare these points in the latent space \mathbb{L} . We chose operations for m_2 such that it forms a metric $\forall \tilde{\mathbf{x}}, \tilde{\mathbf{y}} \in \mathbb{L}$. Since m_1 always maps to \mathbb{L} , this means m has the properties (1), (2), and (3) on \mathbb{I} for any possible mapping m_1 , i.e., only a metric on \mathbb{L} is required. To achieve property (4), m_1 would need to be injective, but the compression of typical feature extractors precludes this. However, if m_1 is deterministic $m(\mathbf{x}, \mathbf{x}) = 0 \forall \mathbf{x} \in \mathbb{I}$ is still fulfilled since identical inputs result in the same point in latent space and thus a distance of zero. More details for this proof can be found in App. A.

3.1. Base Network

The sole purpose of the base network (Fig. 2, in purple) is to extract feature maps from both inputs. The Siamese architecture implies that the weights of the base network are shared for both inputs, meaning all feature maps are comparable. We experimented with the feature extracting layers from var-

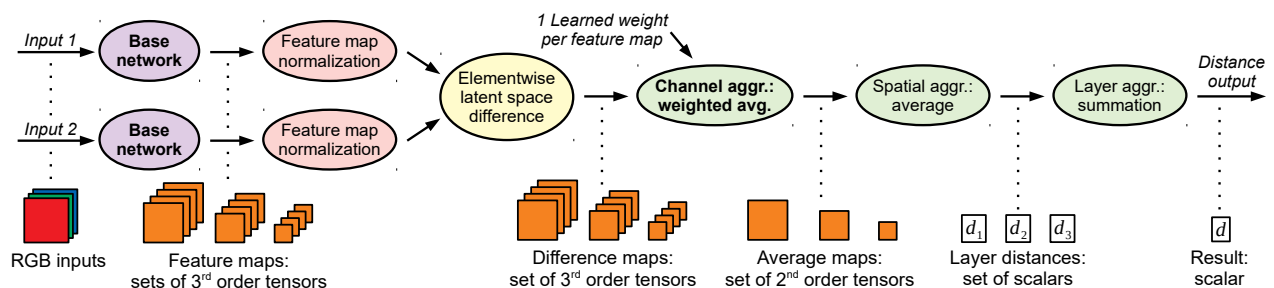


Figure 2. Overview of the proposed distance computation for a simplified base network that contains three layers with four feature maps each in this example. The output shape for every operation is illustrated below the transitions in orange and white. Bold operations are learned, i.e., contain weights influenced by the training process.

ious CNN architectures, such as AlexNet (Krizhevsky et al., 2017), VGG (Simonyan & Zisserman, 2015), SqueezeNet (Iandola et al., 2016), and a fluid flow prediction network (Thurey et al., 2018). We considered three variants of these networks: using the original pre-trained weights, fine-tuning them, or re-training the full networks from scratch. In contrast to typical CNN tasks where only the result of the final output layer is further processed, we make use of the full range of extracted features across the layers of a CNN (see Fig. 2). This implies a slightly different goal compared to regular training: while early features should be general enough to allow for extracting more complex features in deeper layers, this is not their sole purpose. Rather, features in earlier layers of the network can directly participate in the final distance calculation and can yield important cues.

We achieved the best performance for our data sets using a base network architecture with five layers, similar to a reduced AlexNet, that was trained from scratch (see App. B.1). This feature extractor is fully convolutional and thus allows for varying spatial input dimensions, but for comparability to other models we keep the input size constant at 224×224 for our evaluation. In separate tests with interpolated inputs, we found that the metric still works well for scaling factors in the range $[0.5, 2]$.

3.2. Feature Map Normalization

The goal of normalizing the feature maps (Fig. 2, in red) is to transform the extracted features of each layer, which typically have very different orders of magnitude, into comparable ranges. While this task could potentially be performed by the learned weights, we found the normalization to yield improved performance in general.

Let \mathbf{G} denote a 4th order feature tensor with dimensions (g_b, g_c, g_x, g_y) from one layer of the base network. We form a series $\mathbf{G}_0, \mathbf{G}_1, \dots$ for every possible content of this tensor across our training samples. The normalization only happens in the channel dimension, so all following operations accumulate values along the dimension of g_c while keeping $g_b, g_x,$ and g_y constant, i.e., are applied independently of the batch and spatial dimensions. The unit length normalization proposed by Zhang et al., i.e.,

$$\text{norm}_{\text{unit}}(\mathbf{G}) = \mathbf{G} / \|\mathbf{G}\|_2,$$

only considers the current sample. In this case, $\|\mathbf{G}\|_2$ is a 3rd order tensor with the Euclidean norms of \mathbf{G} along the channel dimension. Effectively, this results in a cosine distance, which only measures angles of the latent space vectors. To consider the vector magnitude, the most basic idea is to use the maximum norm of other training samples, and this leads to a global unit length normalization

$$\text{norm}_{\text{global}}(\mathbf{G}) = \mathbf{G} / \max(\|\mathbf{G}_0\|_2, \|\mathbf{G}_1\|_2, \dots).$$

Now, the magnitude of the current sample can be compared to other feature vectors, but this is not robust since the largest feature vector could be an outlier with respect to the typical content. Instead, we individually transform each component of a feature vector with dimension g_c to a standard normal distribution. This is realized by subtracting the mean and dividing by the standard deviation of all features element-wise along the channel dimension as follows:

$$\text{norm}_{\text{dist}}(\mathbf{G}) = \frac{1}{\sqrt{g_c - 1}} \frac{\mathbf{G} - \text{mean}(\mathbf{G}_0, \mathbf{G}_1, \dots)}{\text{std}(\mathbf{G}_0, \mathbf{G}_1, \dots)}.$$

These statistics are computed via a preprocessing step over the training data and stay fixed during training, as we did not observe significant improvements with more complicated schedules such as keeping a running mean. The magnitude of the resulting normalized vectors follows a chi distribution with $k = g_c$ degrees of freedom, but computing its mean $\sqrt{2} \Gamma((k+1)/2) / \Gamma(k/2)$ is expensive¹, especially for larger k . Instead, the mode of the chi distribution $\sqrt{g_c - 1}$ that closely approximates its mean is employed to achieve a consistent average magnitude of about one independently of g_c . As a result, we can measure angles for the latent space vectors and compare their magnitude in the global length distribution across all layers.

3.3. Latent Space Differences

Computing the difference of two latent space representations $\tilde{\mathbf{x}}, \tilde{\mathbf{y}} \in \mathbb{L}$ that consist of all extracted features from the two inputs $\mathbf{x}, \mathbf{y} \in \mathbb{I}$ lies at the core of the metric. This difference operator in combination with the following aggregations has to ensure that the metric properties above are upheld with respect to \mathbb{L} . Thus, the most obvious approach to employ an element-wise difference $\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i \forall i \in \{0, 1, \dots, \dim(\mathbb{L})\}$ is not suitable, as it invalidates non-negativity and symmetry. Instead, exponentiation of an absolute difference via $|\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i|^p$ yields an L^p metric on \mathbb{L} , when combined with the correct aggregation and a p th root. $|\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i|^2$ is used to compute the difference maps (Fig. 2, in yellow), as we did not observe significant differences for other values of p .

Considering the importance of comparing the extracted features, this simple feature difference does not seem optimal. Rather, one can imagine that improvements in terms of comparing one set of feature activations could lead to overall improvements for derived metrics. We investigated replacing these operations with a pre-trained CNN-based metric for each feature map. This creates a recursive process or “meta-metric” that reformulates the initial problem of learning input similarities in terms of learning feature space similarities. However, as detailed in App. B.3, we did not find any substantial improvements with this recursive approach. This implies that once a large enough number of expressive

¹ Γ denotes the gamma function for factorials

features is available for comparison, the in-place difference of each feature is sufficient to compare two inputs.

3.4. Aggregations

The subsequent aggregation operations (Fig. 2, in green) are applied to the difference maps to compress the contained per feature differences along the different dimensions into a single distance value. A simple summation in combination with an absolute difference $|\hat{x}_i - \hat{y}_i|$ above leads to an L^1 distance on the latent space \mathbb{L} . Similarly, we can show that average or learned weighted average operations are applicable too (see App. A). In addition, using a p -th power for the latent space difference requires a corresponding root operation after all aggregations, to ensure the metric properties with respect to \mathbb{L} .

To aggregate the difference maps along the channel dimension, we found the weighted average proposed by Zhang et al. to work very well. Thus, we use one learnable weight to control the importance of a feature. The weight is a multiplier for the corresponding difference map before summation along the channel dimension, and is clamped to be non-negative. A negative weight would mean that a larger difference in this feature produces a smaller overall distance, which is not helpful. For regularization, the learned aggregation weights utilize dropout during training, i.e., are randomly set to zero with a probability of 50%. This ensures that the network cannot rely on single features only, but has to consider multiple features for a more stable evaluation.

For spatial and layer aggregation, functions such as a summation or averaging are sufficient and generally interchangeable. We experimented with more intricate aggregation functions, e.g., by learning a spatial average or determining layer importance weights dynamically from the inputs. When the base network is fixed and the metric only has very few trainable weights, this did improve the overall performance. But, with a fully trained base network, the feature extraction seems to automatically adopt these aspects making a more complicated aggregation unnecessary.

4. Data Generation and Training

Similarity data sets for natural images typically rely on changing already existing images with distortions, noise, or other operations and assigning ground truth distances according to the strength of the operation. Since we can control the data creation process for numerical simulations directly, we can generate large amounts of simulation data with increasing dissimilarities by altering the parameters used for the simulations. As a result, the data contains more information about the nature of the problem, i.e., which changes of the data distribution should lead to increased distances, than by applying modifications as a post-process.

4.1. Data Generation

Given a set of model equations, e.g., a PDE from fluid dynamics, typical solution methods consist of a solver that, given a set of boundary conditions, computes discrete approximations of the necessary differential operators. The discretized operators and the boundary conditions typically contain problem dependent parameters, which we collectively denote with $p_0, p_1, \dots, p_i, \dots$ in the following. We only consider time dependent problems, and our solvers start with initial conditions at t_0 to compute a series of time steps t_1, t_2, \dots until a target point in time (t_t) is reached. At that point, we obtain a reference output field o_0 from one of the PDE variables, e.g., a velocity.

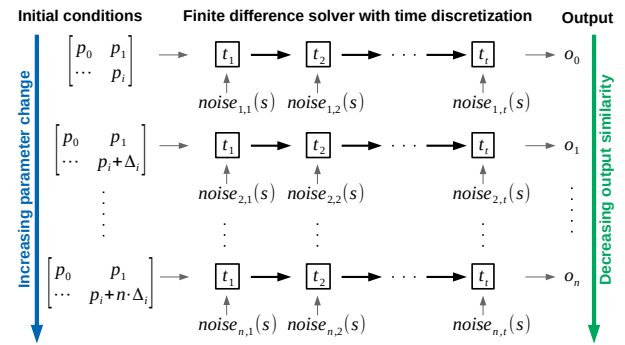


Figure 3. General data generation method from a PDE solver for a time dependent problem. With increasing changes of the initial conditions for a parameter p_i in Δ_i increments, the outputs decrease in similarity. Controlled Gaussian noise is injected in a simulation field of the solver. The difficulty of the learning task can be controlled by scaling Δ_i as well as the noise variance v .

For data generation, we incrementally change a single parameter p_i in n steps $\Delta_i, 2 \cdot \Delta_i, \dots, n \cdot \Delta_i$ to create a series of n outputs o_1, o_2, \dots, o_n . We consider a series obtained in this way to be increasingly different from o_0 . To create natural variations of the resulting data distributions, we add Gaussian noise fields with zero mean and adjustable variance v to an appropriate simulation field such as a velocity. This noise allows us to generate a large number of varied data samples for a single simulation parameter p_i . Furthermore, v serves as an additional parameter that can be varied in isolation to observe the same simulation with different levels of interference. This is similar in nature to numerical errors introduced by discretization schemes. These perturbations enlarge the space covered by the training data, and we found that training networks with suitable noise levels improves robustness as we will demonstrate below. The process for data generation is summarized in Fig. 3.

As PDEs can model extremely complex and chaotic behaviour, there is no guarantee that the outputs always exhibit increasing dissimilarity with the increasing parameter change. This behaviour is what makes the task of similar-

ity assessment so challenging. Even if the solutions are essentially chaotic, their behaviour is not arbitrary but rather governed by the rules of the underlying PDE. For our data set, we choose the following range of representative PDEs: We include a pure Advection-Diffusion model (AD), and Burger’s equation (BE) which introduces an additional viscosity term. Furthermore, we use the full Navier-Stokes equations (NSE), which introduce a conservation of mass constraint. When combined with a deterministic solver and a suitable parameter step size, all these PDEs exhibit chaotic behaviour at small scales, and the medium to large scale characteristics of the solutions shift smoothly with increasing changes of the parameters p_i .

The noise amplifies the chaotic behaviour to larger scales and provides a controlled amount of perturbations for the data generation. This lets the network learn about the nature of the chaotic behaviour of PDEs without overwhelming it with data where patterns are not observable anymore. The latter can easily happen when Δ_i or v grow too large and produce essentially random outputs. Instead, we specifically target solutions that are difficult to evaluate in terms of a shallow metric. We heuristically select the smallest v and a suitable Δ_i such that the ordering of several random output samples with respect to their L^2 difference drops below a correlation value of 0.8. For the chosen PDEs, v was small enough to avoid deterioration of the physical behaviour especially due to the diffusion terms, but different means of adjusting the difficulty may be necessary for other data.

4.2. Training

For training, the 2D scalar fields from the simulations were augmented with random flips, 90° rotations, and cropping to obtain an input size of 224×224 every time they are used. Identical augmentations were applied to each field of one given sequence to ensure comparability. Afterwards, each input sequence is collectively normalized to the range $[0, 255]$. To allow for comparisons with image metrics and provide the possibility to compare color data and full velocity fields during inference, the metric uses three input channels. During training, the scalar fields are duplicated to each channel after augmentation. Unless otherwise noted, networks were trained with a batch size of 1 for 40 epochs with the Adam optimizer using a learning rate of 10^{-5} . To evaluate the trained networks on validation and test inputs, only a bilinear resizing and the normalization step is applied.

5. Correlation Loss Function

The central goal of our networks is to identify relative differences of input pairs produced via numerical simulations. Thus, instead of employing a loss that forces the network to only infer given labels or distance values, we train our networks to infer the ordering of a given sequence of simula-

tion outputs o_0, o_1, \dots, o_n . We propose to use the Pearson correlation coefficient (see Pearson, 1920), which yields a value in $[-1, 1]$ that measures the linear relationship between two distributions. A value of 1 implies that a linear equation describes their relationship perfectly. We compute this coefficient for a full series of outputs such that the network can learn to extract features that arrange this data series in the correct ordering. Each training sample of our network consists of every possible pair from the sequence o_0, o_1, \dots, o_n and the corresponding ground truth distance distribution $\mathbf{c} \in [0, 1]^{0.5(n+1)n}$ representing the parameter change from the data generation. For a distance prediction $\mathbf{d} \in [0, \infty)^{0.5(n+1)n}$ of our network for one sample, we compute the loss with:

$$L(\mathbf{c}, \mathbf{d}) = \lambda_1(\mathbf{c} - \mathbf{d})^2 + \lambda_2 \left(1 - \frac{(\mathbf{c} - \bar{\mathbf{c}}) \cdot (\mathbf{d} - \bar{\mathbf{d}})}{\|\mathbf{c} - \bar{\mathbf{c}}\|_2 \|\mathbf{d} - \bar{\mathbf{d}}\|_2} \right) \quad (5)$$

Here, the mean of a distance vector is denoted by $\bar{\mathbf{c}}$ and $\bar{\mathbf{d}}$ for ground truth and prediction, respectively. The first part of the loss is a regular MSE term, which minimizes the difference between predicted and actual distances. The second part is the Pearson correlation coefficient, which is inverted such that the optimization results in a maximization of the correlation. As this formulation depends on the length of the input sequence, the two terms are scaled to adjust their relative influence with λ_1 and λ_2 . For the training, we chose $n = 10$ variations for each reference simulation. If n should vary during training, the influence of both terms needs to be adjusted accordingly. We found that scaling both terms to a similar order of magnitude worked best in our experiments.

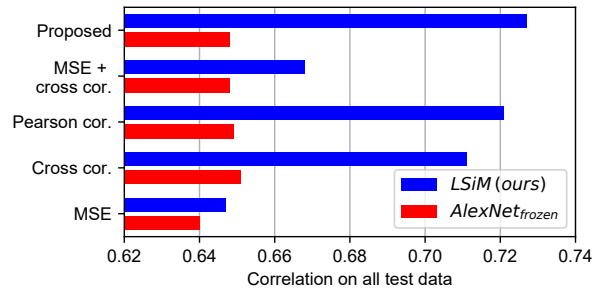


Figure 4. Performance comparison on our test data of the proposed approach (*LSiM*) and a smaller model (*AlexNet_{frozen}*) for different loss functions on the y-axis.

In Fig. 4, we investigate how the proposed loss function compares to other commonly used loss formulations for our full network and a pre-trained network, where only aggregation weights are learned. The performance is measured via Spearman’s rank correlation of predicted against ground truth distances on our combined test data sets. This is comparable to the All column in Tab. 1 and described in more

detail in Section 6.2. In addition to our full loss function, we consider a loss function that replaces the Pearson correlation with a simpler cross-correlation $(\mathbf{c} \cdot \mathbf{d}) / (\|\mathbf{c}\|_2 \|\mathbf{d}\|_2)$. We also include networks trained with only the MSE or only the correlation terms for each of the two variants.

A simple MSE loss yields the worst performance for both evaluated models. Using any correlation based loss function for the *AlexNet_{frozen}* metric (see Section 6.2) improves the results, but there is no major difference due to the limited number of only 1152 trainable weights. For *LSiM*, the proposed combination of MSE loss with the Pearson correlation performs better than using cross-correlation or only isolated Pearson correlation. Interestingly, combining cross correlation with MSE yields worse results than cross correlation by itself. This is caused by the cross correlation term influencing absolute distance values, which potentially conflicts with the MSE term. For our loss, the Pearson correlation only handles the relative ordering while the MSE deals with the absolute distances, leading to better inferred distances.

6. Results

In the following, we will discuss how the data generation approach was employed to create a large range of training and test data from different PDEs. Afterwards, the proposed metric is compared to other metrics, and its robustness is evaluated with several external data sets.

6.1. Data Sets

We created four training (*Smo*, *Liq*, *Adv* and *Bur*) and two test data sets (*LiqN* and *AdvD*) with ten parameter steps for each reference simulation. Based on two 2D NSE solvers, the smoke and liquid simulation training sets (*Smo* and *Liq*) add noise to the velocity field and feature varied initial conditions such as fluid position or obstacle properties, in addition to variations of buoyancy and gravity forces. The two other training sets (*Adv* and *Bur*) are based on 1D solvers for AD and BE, concatenated over time to form a 2D result. In both cases, noise was injected into the velocity field, and the varied parameters are changes to the field initialization and forcing functions.

For the test data set, we substantially change the data distribution by injecting noise into the density instead of the velocity field for AD simulations to obtain the *AdvD* data set and by including background noise for the velocity field of a liquid simulation (*LiqN*). In addition, we employed three more test sets (*Sha*, *Vid*, and *TID*) created without PDE models to explore the generalization for data far from our training data setup. We include a shape data set (*Sha*) that features multiple randomized moving rigid shapes, a video data set (*Vid*) consisting of frames from random video footage, and *TID2013* (Ponomarenko et al., 2015) as a perceptual image data set (*TID*). Below, we additionally list a combined correlation score (*All*) for all test sets apart from *TID*, which is excluded due to its different structure. Examples for each data set are shown in Fig. 5 and generation details with further samples can be found in App. D.

6.2. Performance Evaluation

To evaluate the performance of a metric on a data set, we first compute the distances from each reference simulation to all corresponding variations. Then, the predicted and the ground truth distance distributions over all samples are combined and compared using Spearman’s rank correlation coefficient (see Spearman, 1904). It is similar to the Pearson correlation, but instead it uses ranking variables, i.e., measures monotonic relationships of distributions.

The top part of Tab. 1 shows the performance of the shallow metrics L^2 and *SSIM* as well as the *LPIPS* metric (Zhang et al., 2018) for all our data sets. The results clearly show that shallow metrics are not suitable to compare the samples in our data set and only rarely achieve good correlation values. The perceptual *LPIPS* metric performs better in general and outperforms our method on the image data sets *Vid* and *TID*. This is not surprising as *LPIPS* is specifically trained for such images. For most of the simulation data sets, however, it performs significantly worse than for the image content. The last row of Tab. 1 shows the results of our *LSiM* model with a very good performance across all data sets and no negative outliers. Note that although it was not trained with any natural image content, it still performs well for the image test sets.

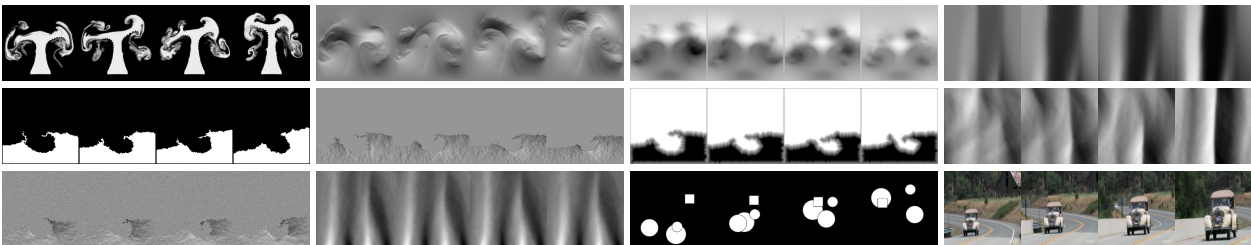
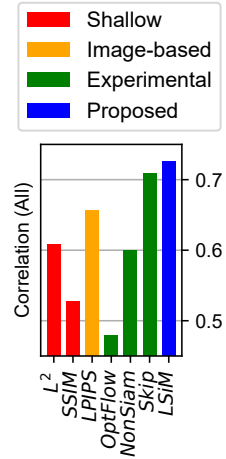


Figure 5. Samples from our data sets. For each subset the reference is on the left, and three variations in equal parameter steps follow. From left to right and top to bottom: *Smo* (density, velocity, and pressure), *Adv* (density), *Liq* (flags, velocity, and levelset), *Bur* (velocity), *LiqN* (velocity), *AdvD* (density), *Sha* and *Vid*.

Table 1. Performance comparison of existing metrics (top block), experimental designs (middle block), and variants of the proposed method (bottom block) on validation and test data sets measured in terms of Spearman’s rank correlation coefficient of ground truth against predicted distances. **Bold+underlined** values show the best performing metric for each data set, **bold** values are within a 0.01 error margin of the best performing, and *italic* values are 0.2 or more below the best performing. On the right, a visualization of the combined test data results is shown for selected models.

Metric	Validation data sets				Test data sets					
	Smo	Liq	Adv	Bur	TID	LiqN	AdvD	Sha	Vid	All
L^2	0.66	0.80	0.74	0.62	0.82	0.73	0.57	<i>0.58</i>	0.79	0.61
SSIM	0.69	0.73	0.77	0.71	0.77	<i>0.26</i>	0.69	<i>0.46</i>	0.75	<i>0.53</i>
LPIPS v0.1.	0.63	0.68	0.68	0.72	0.86	<i>0.50</i>	0.62	0.84	0.83	0.66
<i>AlexNet</i> _{random}	0.63	0.69	0.69	0.66	0.82	0.64	0.65	<i>0.67</i>	0.81	0.65
<i>AlexNet</i> _{frozen}	0.66	0.70	0.69	0.71	0.85	<i>0.40</i>	0.62	0.87	0.84	0.65
Optical flow	0.62	<i>0.57</i>	<i>0.36</i>	<i>0.37</i>	<i>0.55</i>	<i>0.49</i>	<i>0.28</i>	<i>0.61</i>	0.75	<i>0.48</i>
Non-Siamese	0.77	0.85	0.78	0.74	<i>0.65</i>	0.81	0.64	<i>0.25</i>	0.80	0.60
<i>Skip</i> _{from scratch}	0.79	0.83	0.80	0.74	0.85	0.78	0.61	0.78	0.83	0.71
<i>LSiM</i> _{noiseless}	0.77	0.77	0.76	0.72	0.85	0.62	0.58	0.86	0.82	0.68
<i>LSiM</i> _{strong noise}	0.65	<i>0.65</i>	0.67	0.69	0.84	<i>0.39</i>	0.54	0.89	0.82	0.64
<i>LSiM</i> (ours)	0.78	0.82	0.79	0.75	0.86	0.79	0.58	0.88	0.81	0.73



The middle block of Tab. 1 contains several interesting variants (more details can be found in App. B): *AlexNet*_{random} and *AlexNet*_{frozen} are small models, where the base network is the original AlexNet with pre-trained weights. *AlexNet*_{random} contains purely random aggregation weights without training, whereas *AlexNet*_{frozen} only has trainable weights for the channel aggregation and therefore lacks the flexibility to fully adjust to the data distribution of the numerical simulations. The random model performs surprisingly well in general, pointing to powers of the underlying Siamese CNN architecture.

Recognizing that many PDEs include transport phenomena, we investigated optical flow (Horn & Schunck, 1981) as a means to compute motion from field data. For the *Optical flow* metric, we used FlowNet2 (Ilg et al., 2016) to bidirectionally compute the optical flow field between two inputs and aggregate it to a single distance value by summing all flow vector magnitudes. On the data set Vid that is similar to the training data of FlowNet2, it performs relatively well, but in most other cases it performs poorly. This shows that computing a simple warping from one input to the other is not enough for a stable metric although it seems like an intuitive solution. A more robust metric needs the knowledge of the underlying features and their changes to generalize better to new data.

To evaluate whether a Siamese architecture is really beneficial, we used a *Non-Siamese* architecture that directly predicts the distance from both stacked inputs. For this purpose, we employed a modified version of AlexNet that reduces the weights of the feature extractor by 50% and of the remaining layers by 90%. As expected, this metric

works great on the validation data but has huge problems with generalization, especially on TID and Sha. In addition, even simple metric properties such as symmetry are no longer guaranteed because this architecture does not have the inherent constraints of the Siamese setup. Finally, we experimented with multiple fully trained base networks. As re-training existing feature extractors only provided small improvements, we used a custom base network with skip connections for the *Skip*_{from scratch} metric. Its results already come close to the proposed approach on most data sets.

The last block in Tab. 1 shows variants of the proposed approach trained with varied noise levels. This inherently changes the difficulty of the data. Hence, *LSiM*_{noiseless} was trained with relatively simple data without perturbations, whereas *LSiM*_{strong noise} was trained with strongly varying data. Both cases decrease the capabilities of the trained model on some of the validation and test sets. This indicates that the network needs to see a certain amount of variation at training time in order to become robust, but overly large changes hinder the learning of useful features (also see App. C).

6.3. Evaluation on Real-World Data

To evaluate the generalizing capabilities of our trained metric, we turn to three representative and publicly available data sets of captured and simulated real-world phenomena, namely buoyant flows, turbulence, and weather. For the former, we make use of the *ScalarFlow* data set (Eckert et al., 2019), which consists of captured velocities of buoyant scalar transport flows. Additionally, we include velocity data from the Johns Hopkins Turbulence Database (*JHTDB*)

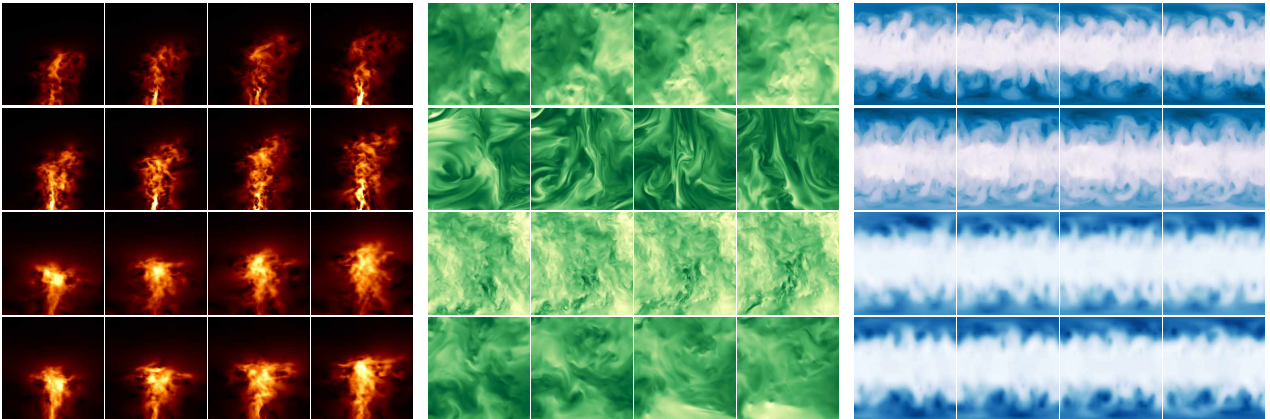


Figure 6. Examples from three real-world data repositories used for evaluation, visualized via color-mapping. Each block features four different sequences (rows) with frames in equal temporal or spatial intervals. Left: *ScalarFlow* – captured buoyant volumetric transport flows using the z-slice (top two) and z-mean (bottom two). Middle: *JHTDB* – four different turbulent DNS simulations. Right: *WeatherBench* – weather data consisting of temperature (top two) and geopotential (bottom two).

(Perlman et al., 2007), which represents direct numerical simulations of fully developed turbulence. As a third case, we use scalar temperature and geopotential fields from the *WeatherBench* repository (Rasp et al., 2020), which contains global climate data on a Cartesian latitude-longitude grid of the earth. Visualizations of this data via color-mapping the scalar fields or velocity magnitudes are shown in Fig. 6.

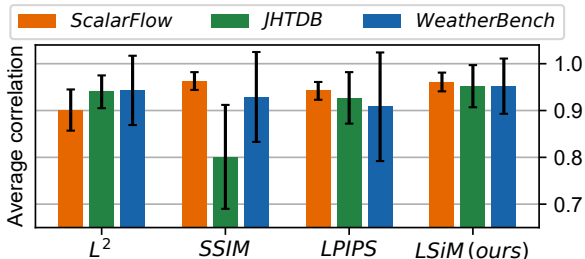


Figure 7. Spearman correlation values for multiple metrics on data from three repositories. Shown are mean and standard deviation over different temporal or spatial intervals used to create sequences.

For the results in Fig. 7, we extracted sequences of frames with fixed temporal and spatial intervals from each data set to obtain a ground truth ordering. Six different interval spacings for every data source are employed, and all velocity data is split by component. We then measure how well different metrics recover the original ordering in the presence of the complex changes of content, driven by the underlying physical processes. The *LSiM* model outlined in previous sections was used for inference without further changes.

Every metric is separately evaluated (see Section 6.2) for the six interval spacings with 180-240 sequences each. For *ScalarFlow* and *WeatherBench*, the data was additionally partitioned by z-slice or z-mean and temperature or geopo-

tential respectively, leading to twelve evaluations. Fig. 7 shows the mean and standard deviation of the resulting correlation values. Despite never being trained on any data from these data sets, *LSiM* recovers the ordering of all three cases with consistently high accuracy. It yields averaged correlations of 0.96 ± 0.02 , 0.95 ± 0.05 , and 0.95 ± 0.06 for *ScalarFlow*, *JHTDB*, and *WeatherBench*, respectively. The other metrics show lower means and higher uncertainty. Further details and results for the individual evaluations can be found in App. E.

7. Conclusion

We have presented the *LSiM* metric to reliably and robustly compare outputs from numerical simulations. Our method significantly outperforms existing shallow metric functions and provides better results than other learned metrics. We demonstrated the usefulness of the correlation loss, showed the benefits of a controlled data generation environment, and highlighted the stability of the obtained metric for a range of real-world data sets.

Our trained *LSiM* metric has the potential to impact a wide range of fields, including the fast and reliable accuracy assessment of new simulation methods, robust optimizations of parameters for reconstructions of observations, and guiding generative models of physical systems. Furthermore, it will be highly interesting to evaluate other loss functions, e.g., mutual information (Bachman et al., 2019) or contrastive predictive coding (Hénaff et al., 2019), and combinations with evaluations from perceptual studies (Um et al., 2019). We also plan to evaluate our approach for an even larger set of PDEs as well as for 3D and 4D data sets. Especially, turbulent flows are a highly relevant and interesting area for future work on learned evaluation metrics.

Acknowledgements

This work was supported by the ERC Starting Grant *realFlow* (StG-2015-637014). We would like to thank Stephan Rasp for preparing the *WeatherBench* data and all reviewers for helping to improve this work.

References

- Agrawal, P., Carreira, J., and Malik, J. Learning to see by moving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 37–45, 2015. doi:10.1109/ICCV.2015.13.
- Amirshahi, S. A., Pedersen, M., and Yu, S. X. Image Quality Assessment by Comparing CNN Features between Images. *Journal of Imaging Science and Technology*, 60(6), 2016. doi:10.2352/J.ImagingSci.Technol.2016.60.6.060410.
- Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. *CoRR*, abs/1906.00910, 2019. URL <http://arxiv.org/abs/1906.00910>.
- Bell, S. and Bala, K. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics*, 34(4):98:1–98:10, 2015. doi:10.1145/2766959.
- Benajiba, Y., Sun, J., Zhang, Y., Jiang, L., Weng, Z., and Biran, O. Siamese networks for semantic pattern similarity. *CoRR*, abs/1812.06604, 2018. URL <http://arxiv.org/abs/1812.06604>.
- Berardino, A., Balle, J., Laparra, V., and Simoncelli, E. Eigen-Distortions of Hierarchical Representations. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, volume 30, 2017. URL <http://arxiv.org/abs/1710.02266>.
- Bertinetto, L., Valmadre, J., Henriques, J. F., Vedaldi, A., and Torr, P. H. S. Fully-Convolutional Siamese Networks for Object Tracking. In *Computer Vision - ECCV 2016 Workshops, PT II*, volume 9914, pp. 850–865, 2016. doi:10.1007/978-3-319-48881-3_56.
- Bosse, S., Maniry, D., Mueller, K.-R., Wiegand, T., and Samek, W. Neural Network-Based Full-Reference Image Quality Assessment. In *2016 Picture Coding Symposium (PCS)*, 2016. doi:10.1109/PCS.2016.7906376.
- Chandar, S., Khapra, M. M., Larochelle, H., and Ravindran, B. Correlational neural networks. *Neural Computation*, 28(2):257–285, 2016. doi:10.1162/NECO_a.00801.
- Chu, M. and Thuerey, N. Data-Driven Synthesis of Smoke Flows with CNN-based Feature Descriptors. *ACM Transactions on Graphics*, 36(4):69:1–69:14, 2017. doi:10.1145/3072959.3073643.
- Dosovitskiy, A. and Brox, T. Generating Images with Perceptual Similarity Metrics based on Deep Networks. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, volume 29, 2016. URL <http://arxiv.org/abs/1602.02644>.
- Eckert, M.-L., Um, K., and Thuerey, N. Scalarflow: A large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics*, 38(6), 2019. doi:10.1145/3355089.3356545.
- Haben, S., Ward, J., Greetham, D. V., Singleton, C., and Grindrod, P. A new error measure for forecasts of household-level, high resolution electrical energy consumption. *International Journal of Forecasting*, 30(2): 246–256, 2014. doi:10.1016/j.ijforecast.2013.08.002.
- Hanif, M. S. Patch match networks: Improved two-channel and Siamese networks for image patch matching. *Pattern Recognition Letters*, 120:54–61, 2019. doi:10.1016/j.patrec.2019.01.005.
- He, H., Chen, M., Chen, T., Li, D., and Cheng, P. Learning to match multitemporal optical satellite images using multi-support-patches Siamese networks. *Remote Sensing Letters*, 10(6):516–525, 2019. doi:10.1080/2150704X.2019.1577572.
- Hénaff, O. J., Razavi, A., Doersch, C., Eslami, S. M. A., and van den Oord, A. Data-efficient image recognition with contrastive predictive coding. *CoRR*, abs/1905.09272, 2019. URL <http://arxiv.org/abs/1905.09272>.
- Horn, B. K. and Schunck, B. G. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981. doi:10.1016/0004-3702(81)90024-2.
- Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016. URL <http://arxiv.org/abs/1602.07360>.
- Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. FlowNet 2.0: Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925, 2016. URL <http://arxiv.org/abs/1612.01925>.
- Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Computer Vision - ECCV 2016, PT II*, volume 9906, pp. 694–711, 2016. doi:10.1007/978-3-319-46475-6_43.

- Jolliffe, I. T. and Stephenson, D. B. *Forecast verification: a practitioner's guide in atmospheric science*. John Wiley & Sons, 2012. doi:10.1002/9781119960003.
- Kang, L., Ye, P., Li, Y., and Doermann, D. Convolutional Neural Networks for No-Reference Image Quality Assessment. In *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1733–1740, 2014. doi:10.1109/CVPR.2014.224.
- Keil, C. and Craig, G. C. A displacement and amplitude score employing an optical flow technique. *Weather and Forecasting*, 24(5):1297–1308, 2009. doi:10.1175/2009WAF2222247.1.
- Kim, J. and Lee, S. Deep Learning of Human Visual Sensitivity in Image Quality Assessment Framework. In *30TH IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017)*, pp. 1969–1977, 2017. doi:10.1109/CVPR.2017.213.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. doi:10.1145/3065386.
- Larson, E. C. and Chandler, D. M. Most apparent distortion: full-reference image quality assessment and the role of strategy. *Journal of Electronic Imaging*, 19(1), 2010. doi:10.1117/1.3267105.
- Lin, Z., Hahn, T. S., Lee, W., Tang, W. M., and White, R. B. Turbulent transport reduction by zonal flows: Massively parallel simulations. *Science*, 281(5384):1835–1837, 1998. doi:10.1126/science.281.5384.1835.
- Liu, X., Pedersen, M., and Hardeberg, J. Y. CID:IQ - A New Image Quality Database. In *Image and Signal Processing, ICISP 2014*, volume 8509, pp. 193–202, 2014. doi:10.1007/978-3-319-07998-1_22.
- Moin, P. and Mahesh, K. Direct numerical simulation: a tool in turbulence research. *Annual review of fluid mechanics*, 30(1):539–578, 1998. doi:10.1146/annurev.fluid.30.1.539.
- Oberkampf, W. L., Trucano, T. G., and Hirsch, C. Verification, validation, and predictive capability in computational engineering and physics. *Applied Mechanics Reviews*, 57:345–384, 2004. doi:10.1115/1.1767847.
- Pearson, K. Notes on the History of Correlation. *Biometrika*, 13(1):25–45, 1920. doi:10.1093/biomet/13.1.25.
- Perlman, E., Burns, R., Li, Y., and Meneveau, C. Data exploration of turbulence simulations using a database cluster. In *SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, pp. 1–11, 2007. doi:10.1145/1362622.1362654.
- Pitsch, H. Large-eddy simulation of turbulent combustion. *Annu. Rev. Fluid Mech.*, 38:453–482, 2006. doi:10.1146/annurev.fluid.38.050304.092133.
- Ponomarenko, N., Jin, L., Ieremeiev, O., Lukin, V., Egiazarian, K., Astola, J., Vozel, B., Chehdi, K., Carli, M., Battisti, F., and Kuo, C. C. J. Image database TID2013: Peculiarities, results and perspectives. *Signal Processing-Image Communication*, 30:57–77, 2015. doi:10.1016/j.image.2014.10.009.
- Prashnani, E., Cai, H., Mostofi, Y., and Sen, P. Pieapp: Perceptual image-error assessment through pairwise preference. *CoRR*, abs/1806.02067, 2018. URL <http://arxiv.org/abs/1806.02067>.
- Rasp, S., Dueben, P., Scher, S., Weyn, J., Mouatadid, S., and Thuerey, N. Weatherbench: A benchmark dataset for data-driven weather forecasting. *CoRR*, abs/2002.00469, 2020. URL <http://arxiv.org/abs/2002.00469>.
- Ruder, M., Dosovitskiy, A., and Brox, T. Artistic style transfer for videos. In *Pattern Recognition*, pp. 26–36, 2016. doi:10.1007/978-3-319-45886-1_3.
- Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. URL <http://arxiv.org/abs/1409.1556>.
- Spearman, C. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. doi:10.2307/1412159.
- Talebi, H. and Milanfar, P. Learned Perceptual Image Enhancement. In *2018 IEEE International Conference on Computational Photography (ICCP)*, 2018a. doi:10.1109/ICCPHOT.2018.8368474.
- Talebi, H. and Milanfar, P. NIMA: Neural Image Assessment. *IEEE Transactions on Image Processing*, 27(8):3998–4011, 2018b. doi:10.1109/TIP.2018.2831899.
- Thuerey, N., Weissenow, K., Mehrotra, H., Mainali, N., Prantl, L., and Hu, X. Well, how accurate is it? A study of deep learning methods for reynolds-averaged navier-stokes simulations. *CoRR*, abs/1810.08217, 2018. URL <http://arxiv.org/abs/1810.08217>.
- Um, K., Hu, X., and Thuerey, N. Perceptual Evaluation of Liquid Simulation Methods. *ACM Transactions on Graphics*, 36(4), 2017. doi:10.1145/3072959.3073633.
- Um, K., Hu, X., Wang, B., and Thuerey, N. Spot the Difference: Accuracy of Numerical Simulations via the Human Visual System. *CoRR*, abs/1907.04179, 2019. URL <http://arxiv.org/abs/1907.04179>.

- Wang, X. and Gupta, A. Unsupervised learning of visual representations using videos. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2794–2802, 2015. doi:10.1109/ICCV.2015.320.
- Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi:10.1109/TIP.2003.819861.
- Wang, Z., Zhang, J., and Xie, Y. L2 Mispronunciation Verification Based on Acoustic Phone Embedding and Siamese Networks. In *2018 11TH International Symposium on Chinese Spoken Language Processing (ISCSLP)*, pp. 444–448, 2018. doi:10.1109/ISCSLP.2018.8706597.
- Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018. doi:10.1109/CVPR.2018.00068.
- Zhang, Y. and Duan, Z. IMINET: Convolutional Semi-Siamese Networks for Sound Search by Vocal Imitation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pp. 304–308, 2017. doi:10.1109/TASLP.2018.2868428.

Appendix: Learning Similarity Metrics for Numerical Simulations

This supplemental document contains an analysis of the proposed metric design with respect to properties of metrics in general (App. A) and details to the used network architectures (App. B). Afterwards, material that deals with the data sets is provided. It contains examples and failure cases for each of the data domains and analyzes the impact of the data difficulty (App. C and D). Next, the evaluation on real-world data is described in more detail (App. E). Finally, we explore additional metric evaluations (App. F) and give an overview on the used notation (App. G).

The source code for using the trained *LSiM* metric and re-training the model from scratch are available at <https://github.com/tum-pbs/LSiM>. This includes the full data sets and the corresponding data generation scripts for the employed PDE solver.

A. Discussion of Metric Properties

To analyze if the proposed method qualifies as a *metric*, it is split in two functions $m_1 : \mathbb{I} \rightarrow \mathbb{L}$ and $m_2 : \mathbb{L} \times \mathbb{L} \rightarrow [0, \infty)$, which operate on the input space \mathbb{I} and the latent space \mathbb{L} . Through flattening elements from the input or latent space into vectors, $\mathbb{I} \simeq \mathbb{R}^a$ and $\mathbb{L} \simeq \mathbb{R}^b$ where a and b are the dimensions of the input data and all feature maps respectively, and both values have a similar order of magnitude. m_1 describes the non-linear function computed by the base network combined with the following normalization and returns a point in the latent space. m_2 uses two points in the latent space to compute a final distance value, thus it includes the latent space difference and the aggregation along the spatial, layer, and channel dimensions. With the Siamese network architecture, the resulting function for the entire approach is

$$m(\mathbf{x}, \mathbf{y}) = m_2(m_1(\mathbf{x}), m_1(\mathbf{y})).$$

The identity of indiscernibles mainly depends on m_1 because, even if m_2 itself guarantees this property, m_1 could still be non-injective, which means it can map different inputs to the same point in latent space $\tilde{\mathbf{x}} = \tilde{\mathbf{y}}$ for $\mathbf{x} \neq \mathbf{y}$. Due to the complicated nature of m_1 , it is difficult to make accurate predictions about the injectivity of m_1 . Each base network layer of m_1 recursively processes the result of the preceding layer with various feature extracting operations. Here, the intuition is that significant changes in the input should produce different feature map results in one or more layers of the network. As very small changes in the input lead to zero valued distances predicted by the CNN (i.e., an

identical latent space for different inputs), m_1 is in practice not injective. In an additional experiment, the proposed architecture was evaluated on about 3500 random inputs from all our data sets, where the CNN received one unchanged and one slightly modified input. The modification consisted of multiple pixel adjustments by one bit (on 8-bit color images) in random positions and channels. When adjusting only a single pixel in the 224×224 input, the CNN predicts a zero valued distance on about 23% of the inputs, but we never observed an input where seven or more changed pixels resulted in a distance of zero in all experiments.

In this context, the problem of numerical errors is important because even two slightly different latent space representations could lead to a result that seems to be zero if the difference vanishes in the aggregation operations or is smaller than the floating point precision. On the other hand, an automated analysis to find points that have a different input but an identical latent space image is a challenging problem and left as future work.

The evaluation of the base network and the normalization is deterministic, and hence $\forall \mathbf{x} : m_1(\mathbf{x}) = m_1(\mathbf{x})$ holds. Furthermore, we know that $m(\mathbf{x}, \mathbf{x}) = 0$ if m_2 guarantees that $\forall m_1(\mathbf{x}) : m_2(m_1(\mathbf{x}), m_1(\mathbf{x})) = 0$. Thus, the remaining properties, i.e., non-negativity, symmetry, and the triangle inequality, only depend on m_2 since for them the original inputs are not relevant, but their respective images in the latent space. The resulting structure with a relaxed identity of indiscernibles is called a *pseudometric*, where $\forall \tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}} \in \mathbb{L}$:

$$m_2(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \geq 0 \quad (1)$$

$$m_2(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = m_2(\tilde{\mathbf{y}}, \tilde{\mathbf{x}}) \quad (2)$$

$$m_2(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \leq m_2(\tilde{\mathbf{x}}, \tilde{\mathbf{z}}) + m_2(\tilde{\mathbf{z}}, \tilde{\mathbf{y}}) \quad (3)$$

$$m_2(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) = 0 \quad (4)$$

Notice that m_2 has to fulfill these properties with respect to the latent space but not the input space. If m_2 is carefully constructed, the metric properties still apply, independently of the actual design of the base network or the feature map normalization.

A first observation concerning m_2 is that if all aggregations were sum operations and the element-wise latent space difference was the absolute value of a difference operation, m_2 would be equivalent to computing the L^1 norm of the difference vector in latent space:

$$m_2^{sum}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \sum_{i=1}^b |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i|.$$

Similarly, adding a square operation to the element-wise distance in the latent space and computing the square root at the very end leads to the L^2 norm of the latent space difference vector. In the same way, it is possible to use any L^p norm with the corresponding operations:

$$m_2^{sum}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \left(\sum_{i=1}^b |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i|^p \right)^{\frac{1}{p}}.$$

In both cases, this forms the metric induced by the corresponding norm, which by definition has all desired properties (1), (2), (3), and (4). If we change all aggregation methods to a weighted average operation, each term in the sum is multiplied by a weight w_i . This is even possible with learned weights, as they are constant at evaluation time if they are clamped to be positive as described above. Now, w_i can be attributed to both inputs by distributivity, meaning each input is element-wise multiplied with a constant vector before applying the metric, which leaves the metric properties untouched. The reason is that it is possible to define new vectors in the same space, equal to the scaled inputs. This renaming trivially provides the correct properties:

$$m_2^{weighted}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \sum_{i=1}^b w_i |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i|,$$

$$\stackrel{w_i > 0}{=} \sum_{i=1}^b |w_i \tilde{\mathbf{x}}_i - w_i \tilde{\mathbf{y}}_i|.$$

Accordingly, doing the same with the L^p norm idea is possible, and each w_i just needs a suitable adjustment before distributivity can be applied, keeping the metric properties once again:

$$m_2^{weighted}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) = \left(\sum_{i=1}^b w_i |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i|^p \right)^{\frac{1}{p}}$$

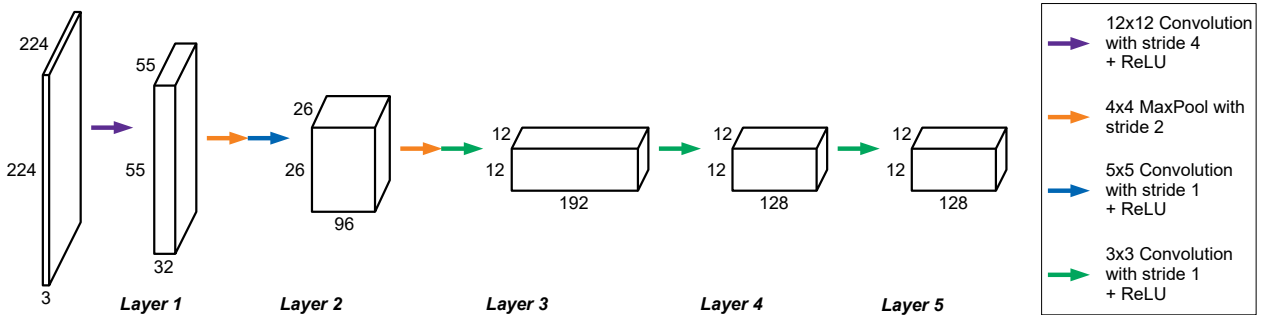


Figure 1. Proposed base network architecture consisting of five layers with up to 192 feature maps that are decreasing in spatial size. It is similar to the feature extractor from AlexNet as identical spatial dimensions for the feature maps are used, but it reduces the number of feature maps for each layer by 50% to have fewer weights.

$$= \left(\sum_{i=1}^b w_i |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i| |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i| \dots |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i| \right)^{\frac{1}{p}}$$

$$= \left(\sum_{i=1}^b w_i^{\frac{1}{p}} |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i| w_i^{\frac{1}{p}} |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i| \dots w_i^{\frac{1}{p}} |\tilde{\mathbf{x}}_i - \tilde{\mathbf{y}}_i| \right)^{\frac{1}{p}},$$

$$\stackrel{w_i > 0}{=} \left(\sum_{i=1}^b |w_i^{\frac{1}{p}} \tilde{\mathbf{x}}_i - w_i^{\frac{1}{p}} \tilde{\mathbf{y}}_i|^p \right)^{\frac{1}{p}}.$$

With these weighted terms for m_2 , it is possible to describe all used aggregations and latent space difference methods. The proposed method deals with multiple higher order tensors instead of a single vector. Thus, the weights w_i additionally depend on constants such as the direction of the aggregations and their position in the latent space tensors. But it is easy to see that mapping a higher order tensor to a vector and keeping track of additional constants still retains all properties in the same way. As a result, the described architecture by design yields a pseudometric that is suitable for comparing simulation data in a way that corresponds to our intuitive understanding of distances.

B. Architectures

The following sections provide details regarding the architecture of the base network and some experimental design.

B.1. Base Network Design

Fig. 1 shows the architecture of the base network for the *LSiM* metric. Its purpose is to extract features from both inputs of the Siamese architecture that are useful for the further processing steps. To maximise the usefulness and to avoid feature maps that show overly similar features, the chosen kernel size and stride of the convolutions are important. Starting with larger kernels and strides means the network has a big receptive field and can consider simple, low-level features in large regions of the input. For the two

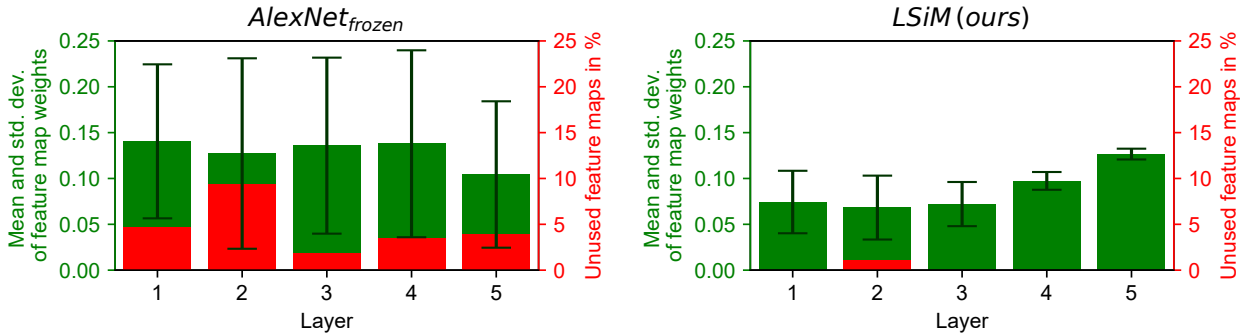


Figure 2. Analysis of the distributions of learned feature map aggregation weights across the base network layers. Displayed is a base network with pre-trained weights (left) in comparison to our method for fully training the base network (right). Note that the percentage of unused feature maps for most layers of our base network is 0%.

following layers, the large strides are replaced by additional MaxPool operations that serve a similar purpose and reduce the spatial size of the feature maps.

For the three final layers, only small convolution kernels and strides are used, but the number of channels is significantly larger than before. These deep feature maps typically contain high-level structures, which are most important to distinguish complex changes in the inputs. Keeping the number of trainable weights as low as possible was an important consideration for this design to prevent overfitting to certain simulation types and increase generality. We explored a weight range by using the same architecture and only scaling the number of feature maps in each layer. The final design shown in Fig. 1 with about 0.62 million weights worked best for our experiments.

In the following, we analyze the contributions of the per-layer features of two different metric networks to highlight differences in terms of how the features are utilized for the distance estimation task. In Fig. 2, our *LSiM* network yields a significantly smaller standard deviation in the learned weights that aggregate feature maps of five layers, compared to a pre-trained base network. This means, all feature maps contribute to establishing the distances similarly, and the aggregation just fine-tunes the relative importance of each feature. In addition, almost all features receive a weight greater than zero, and as a result, more features are contributing to the final distance value.

Employing a fixed pre-trained feature extractor, on the other hand, shows a very different picture: Although the mean across the different network layers is similar, the contributions of different features vary strongly, which is visible in the standard deviation being significantly larger. Furthermore, 2-10% of the feature maps in each layer receive a weight of zero and hence were deemed not useful at all for establishing the distances. This illustrates the usefulness of a targeted network in which all features contribute to the distance inference.

B.2. Feature Map Normalization

In the following, we analyze how the different feature map normalizations discussed in Section 3.2 of the main paper affect the performance of our metric. We compare using no normalization $norm_{none}(\mathbf{G}) = \mathbf{G}$, the unit length normalization via division by the norm of a feature vector $norm_{unit}(\mathbf{G}) = \mathbf{G} / \|\mathbf{G}\|_2$ proposed by Zhang et al., a global unit length normalization $norm_{global}(\mathbf{G}) = \mathbf{G} / \max(\|\mathbf{G}_0\|_2, \|\mathbf{G}_1\|_2, \dots)$ that considers the norm of all feature vectors in the entire training set, and the proposed normalization to a scaled chi distribution

$$norm_{dist}(\mathbf{G}) = \frac{1}{\sqrt{g_c - 1}} \frac{\mathbf{G} - \text{mean}(\mathbf{G}_0, \mathbf{G}_1, \dots)}{\text{std}(\mathbf{G}_0, \mathbf{G}_1, \dots)}.$$

Fig. 3 shows a comparison of these normalization methods on the combined test data. Using no normalization is significantly detrimental to the performance of the metric as succeeding operations cannot reliably compare the features. A unit length normalization of a single sample is already a major improvement since following operations now have a predictable range of values to work with. This corresponds to a cosine distance, which only measures angles of the feature vectors and entirely neglects their length.

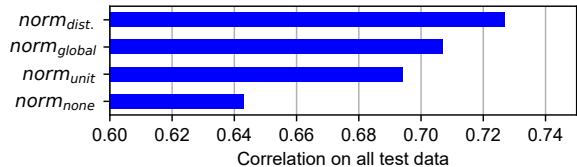


Figure 3. Performance on our test data for different feature map normalization approaches.

Using the maximum norm across all training samples (computed in a pre-processing step and fixed for training) introduces additional information as the network can now compare magnitudes as well. However, this comparison is not stable as the maximum norm can be an outlier with respect to the typical content of the corresponding feature.

The proposed normalization forms a chi distribution by individually transforming each component of the feature vector to a standard normal distribution. Afterwards, scaling with the inverse mode of the chi distribution leads to a consistent average magnitude close to one. It results in the best performing metric since both length and angle of the feature vectors can be reliably compared by the following operations.

B.3. Recursive “Meta-Metric”

Since comparing the feature maps is a central operation of the proposed metric calculations, we experimented with replacing it with an existing CNN-based metric. In theory, this would allow for a recursive, arbitrarily deep network that repeatedly invokes itself: first, the extracted representations of inputs are used and then the representations extracted from the previous representations, etc. In practice, however, using more than one recursion step is currently not feasible due to increasing computational requirements in addition to vanishing gradients.

Fig. 4 shows how our computation method can be modified for a CNN-based latent space difference, instead of an element-wise operation. Here we employ *LPIPS* (Zhang et al., 2018). There are two main differences compared to proposed method. First, the *LPIPS* latent space difference creates single distance values for a pair of feature maps instead of a spatial feature difference. As a result, the following aggregation is a single learned average operation and spatial or layer aggregations are no longer necessary. We also performed experiments with a spatial *LPIPS* version here, but due to memory limitations, these were not successful. Second, the convolution operations in *LPIPS* have a lower limit for spatial resolution, and some feature maps of our base network are quite small (see Fig. 1). Hence, we up-scale the feature maps below the required spatial size of 32×32 using nearest neighbor interpolation.

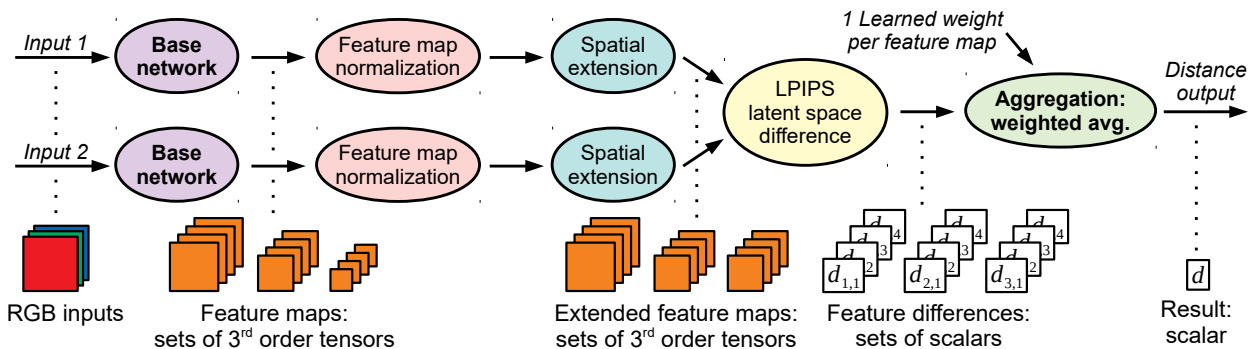


Figure 4. Adjusted distance computation for a *LPIPS*-based latent space difference. To provide sufficiently large inputs for *LPIPS*, small feature maps are spatially enlarged with nearest neighbor interpolation. In addition, *LPIPS* creates scalar instead of spatial differences leading to a simplified aggregation.

On our combined test data, such a metric with a fully trained base network achieves a performance comparable to *AlexNet_{random}* or *AlexNet_{frozen}*.

B.4. Optical Flow Metric

In the following, we describe our approach to compute a metric via optical flow (OF). For an efficient OF evaluation, we employed a pre-trained network (Ilg et al., 2016). From an OF network $f : \mathbb{I} \times \mathbb{I} \rightarrow \mathbb{R}^{i_{max} \times j_{max} \times 2}$ with two input data fields $\mathbf{x}, \mathbf{y} \in I$, we get the flow vector field $f^{\mathbf{xy}}(i, j) = (f_1^{\mathbf{xy}}(i, j), f_2^{\mathbf{xy}}(i, j))^T$, where i and j denote the locations, and f_1 and f_2 denote the components of the flow vectors. In addition, we have a second flow field $f^{\mathbf{yx}}(i, j)$ computed from the reversed input ordering. We can now define a function $m : \mathbb{I} \times \mathbb{I} \rightarrow [0, \infty)$:

$$m(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{i_{max}} \sum_{j=0}^{j_{max}} \sqrt{(f_1^{\mathbf{xy}}(i, j))^2 + (f_2^{\mathbf{xy}}(i, j))^2} + \sqrt{(f_1^{\mathbf{yx}}(i, j))^2 + (f_2^{\mathbf{yx}}(i, j))^2}.$$

Intuitively, this function computes the sum over the magnitudes of all flow vectors in both vector fields. With this definition, it is obvious that $m(\mathbf{x}, \mathbf{y})$ fulfills the metric properties of non-negativity and symmetry (see Eq. (1) and (2)). Under the assumption that identical inputs create a zero flow field, a relaxed identity of indiscernibles holds as well (see Eq. (4)). Compared to the proposed approach, there is no guarantee for the triangle inequality though, thus $m(\mathbf{x}, \mathbf{y})$ only qualifies as a pseudo-semimetric.

Fig. 5 shows flow visualizations on data examples produced by FlowNet2. The metric works relatively well for inputs that are similar to the training data from FlowNet2 such as the shape data example in the top row. For data that provides some outline, e.g., the smoke simulation example in the middle row or also liquid data, the metric does not work

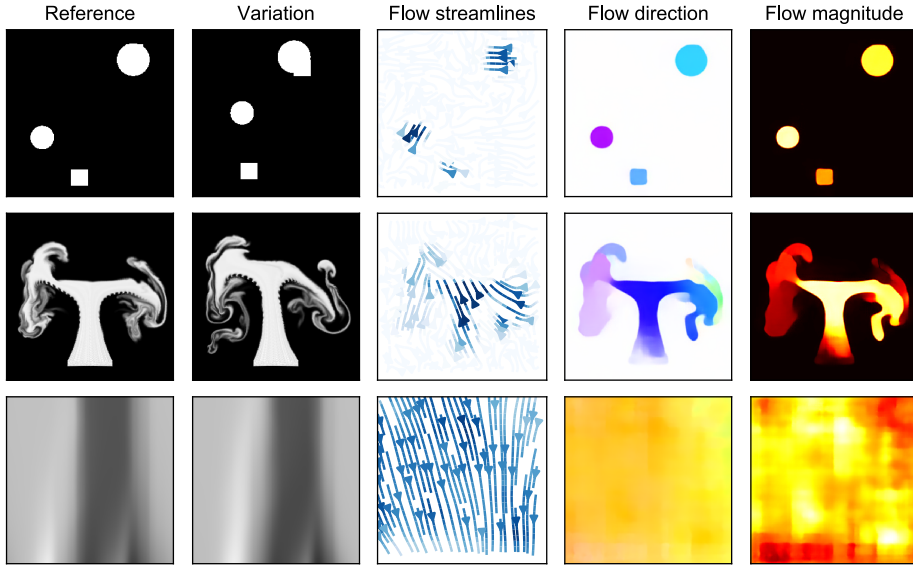


Figure 5. Outputs from FlowNet2 on data examples. The flow streamlines are sparse visualization of the resulting flow field and indicate the direction of the flow by their orientation and its magnitude by their color (darker being larger). The two visualizations on the right show the dense flow field and are color-coded to show the flow direction (blue/yellow: vertical, green/red: horizontal) and the flow magnitude (brighter being larger).

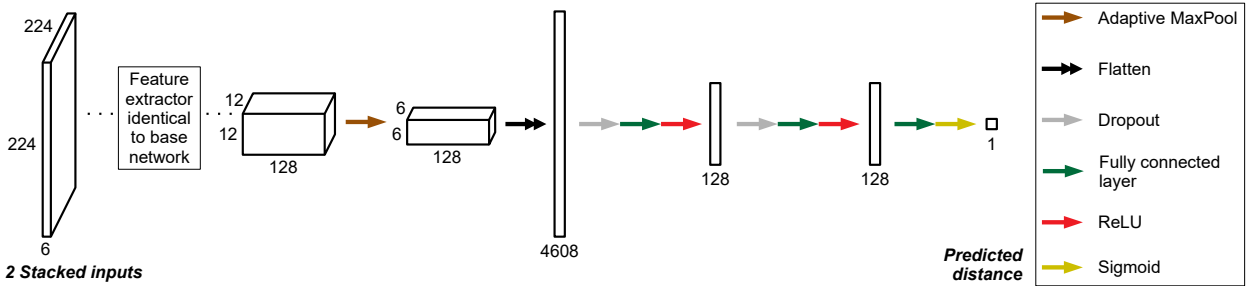


Figure 6. Non-Siamese network architecture with the same feature extractor used in Fig. 1. It uses both stacked inputs and directly predicts the final distance value from the last set of feature maps with several fully connected layers.

as well but still provides a reasonable flow field. However, for full spatial examples such as the Burger’s or Advection-Diffusion cases (see bottom row), the network is no longer able to produce meaningful flow fields. The results are often a very uniform flow with similar magnitude and direction.

B.5. Non-Siamese Architecture

To compute a metric without the Siamese architecture outlined above, we use a network structure with a single output as shown in Fig. 6. Thus, instead of having two identically feature extractors and combining the feature maps, here the distance is directly predicted from the stacked inputs with a single network with about 1.24 million weights. After using the same feature extractor as described in Section B.1, the final set of feature maps is spatially reduced with an adaptive MaxPool operation. Next, the result is flattened, and

three consecutive fully connected layers process the data to form the final prediction. Here, the last activation function is a sigmoid instead of ReLU. The reason is that a ReLU would clamp every negative intermediate value to a zero distance, while a sigmoid compresses the intermediate value to a small distance that is more meaningful than directly clamping it.

In terms of metric properties, this architecture only provides non-negativity (see Eq. (1)) due to the final sigmoid function. All other properties cannot be guaranteed without further constraints. This is the main disadvantage of a non-Siamese network. These issues could be alleviated with specialized training data or by manually adding constraints to the model, e.g., to have some amount of symmetry (see Eq. (2)) and at least a weakened identity of indiscernibles (see Eq. (4)). However, compared to a Siamese network that guarantees

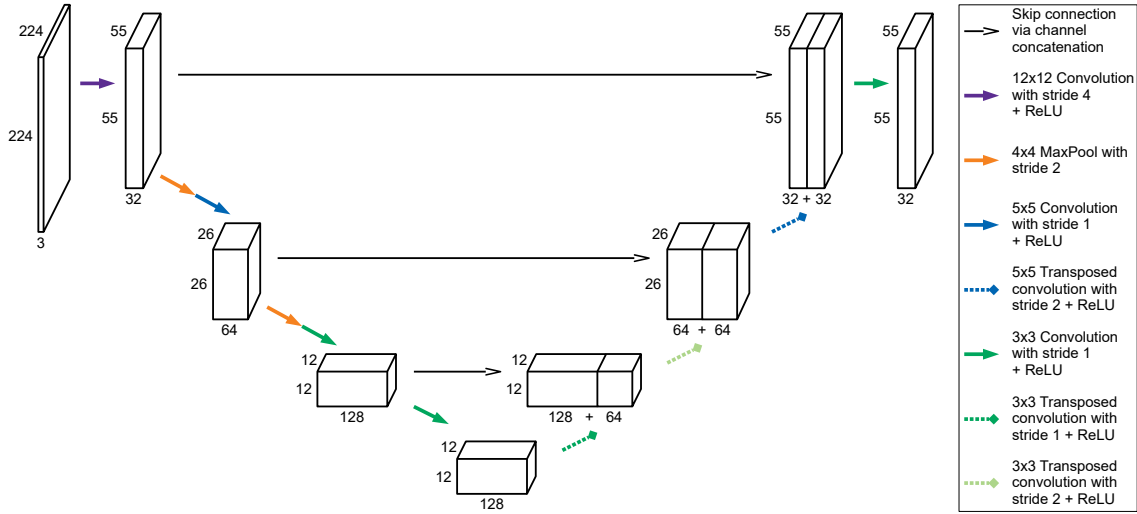


Figure 7. Network architecture with skip connections for better information transport between feature maps. Transposed convolutions are used to upscale the feature maps in the second half of the network to match the spatial size of earlier layers for the skip connections.

them by design, these extensions are clearly sub-optimal. As a result of the missing properties, this network has significant problems with generalization. While it performs well on the training data, the performance noticeably deteriorates for several of the test data sets.

B.6. Skip Connections in Base Network

As explained above, our base network primarily serves as a feature extractor to produce activations that are employed to evaluate a learned metric. In many state-of-the-art methods, networks with skip connections are employed (Ronneberger et al., 2015; He et al., 2016; Huang et al., 2017), as experiments have shown that these connections help to preserve information from the inputs. In our case, the classification “output” of a network such as the AlexNet plays no actual role. Rather, the features extracted along the way are crucial. Hence, skip connections should not improve the inference task for our metrics.

To verify that this is the case, we have included tests with a base network (see Fig. 7) similar to the popular UNet architecture (Ronneberger et al., 2015). For our experiments, we kept the early layers closely in line with the feature extractors that worked well for the base network (see Section B.1). Only the layers in the decoder part have an increased spatial feature map size to accommodate the skip connections. As expected, this network can be used to compute reliable metrics for the input data without negatively affecting the performance. However, as expected, the improvements of skip connections for regular inference tasks do not translate into improvements for the metric calculations.

C. Impact of Data Difficulty

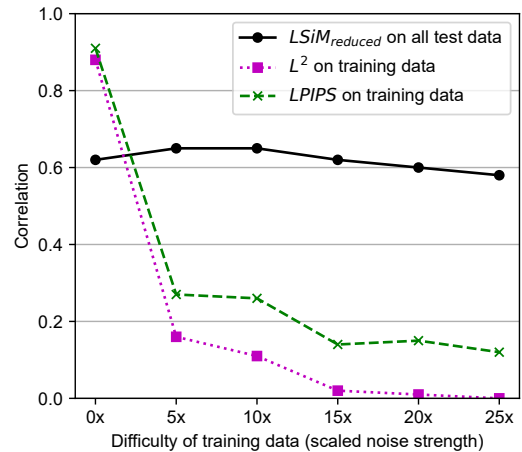


Figure 8. Impact of increasing data difficulty for a reduced training data set. Evaluations on training data for L^2 and $LPIPS$, and the test performance of models trained with the different reduced data sets ($LSiM_{reduced}$) are shown.

We shed more light on the aspect of noise levels and data difficulty via six reduced data sets that consist of a smaller amount of Smoke and Advection-Diffusion data with differently scaled noise strength values. Results are shown in Fig. 8. Increasing the noise level creates more difficult data as shown by the dotted and dashed plots representing the performance of the L^2 and the $LPIPS$ metric on each data set. Both roughly follow an exponentially decreasing function. Each point on the solid line plot is the test result of a reduced $LSiM$ model trained on the data set with the corresponding noise level. Apart from the data, the entire training

setup was identical. This shows that the training process is very robust to the noise, as the result on the test data only slowly decreases for very high noise levels. Furthermore, small amounts of noise improve the generalization compared to the model that was trained without any noise. This is somewhat expected, as a model that never saw noisy data during training cannot learn to extract features which are robust with respect to noise.

D. Data Set Details

In the following sections, the generation of each used data set is described. For each figure showing data samples (consisting of a reference simulation and several variants with a single changing initial parameter), the leftmost image is the reference and the images to the right show the variants in order of increasing parameter change. For the figures 9, 10, 11, and 12, the first subfigure (a) demonstrates that medium and large scale characteristics behave very non-chaotic for simulations without any added noise. They are only included for illustrative purposes and are not used for training. The second and third subfigure (b) and (c) in each case show the training data of *LSiM*, where the large majority of data falls into the category (b) of normal samples that follow the generation ordering, even with more varying behaviour. Category (c) is a small fraction of the training data, and the shown examples are specifically picked to show how the chaotic behaviour can sometimes override the ordering intended by the data generation in the worst case. Occasionally, category (d) is included to show how normal data samples from the test set differ from the training data.

D.1. Navier-Stokes Equations

These equations describe the general behaviour of fluids with respect to advection, viscosity, pressure, and mass conservation. Eq. (5) defines the conservation of momentum, and Eq. (6) constraints the conservation of mass:

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{\nabla P}{\rho} + \nu \nabla^2 u + g, \quad (5)$$

$$\nabla \cdot u = 0. \quad (6)$$

In this context, u is the velocity, P is the pressure the fluid exerts, ρ is the density of the fluid (usually assumed to be constant), ν is the kinematic viscosity coefficient that indicates the thickness of the fluid, and g denotes the acceleration due to gravity. With this PDE, three data sets were created using a smoke and a liquid solver. For all data, 2D simulations were run until a certain step, and useful data fields were exported afterwards.

SMOKE

For the smoke data, a standard Eulerian fluid solver using a preconditioned pressure solver based on the conjugate

gradient method and Semi-Lagrangian advection scheme was employed.

The general setup for every smoke simulation consists of a rectangular smoke source at the bottom with a fixed additive noise pattern to provide smoke plumes with more details. Additionally, there is a downwards directed, spherical force field area above the source, which divides the smoke in two major streams along it. We chose this solution over an actual obstacle in the simulation in order to avoid overfitting to a clearly defined black obstacle area inside the smoke data. Once the simulation reaches a predefined time step, the density, pressure, and velocity fields (separated by dimension) are exported and stored. Some example sequences can be found in Fig. 9. With this setup, the following initial conditions were varied in isolation:

- Smoke buoyancy in x- and y-direction
- Strength of noise added to the velocity field
- Amount of force in x- and y-direction provided by the force field
- Orientation and size of the force field
- Position of the force field in x- and y-direction
- Position of the smoke source in x- and y-direction

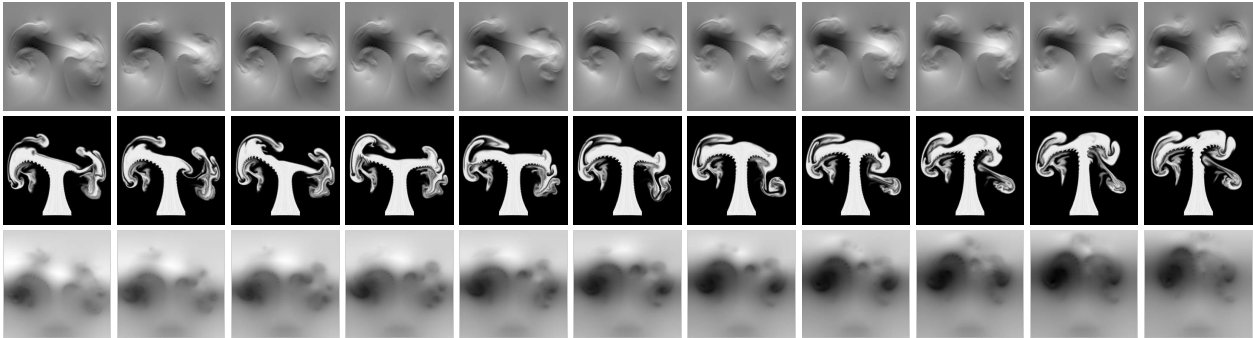
Overall, 768 individual smoke sequences were used for training, and the validation set contains 192 sequences with different initialization seeds.

LIQUID

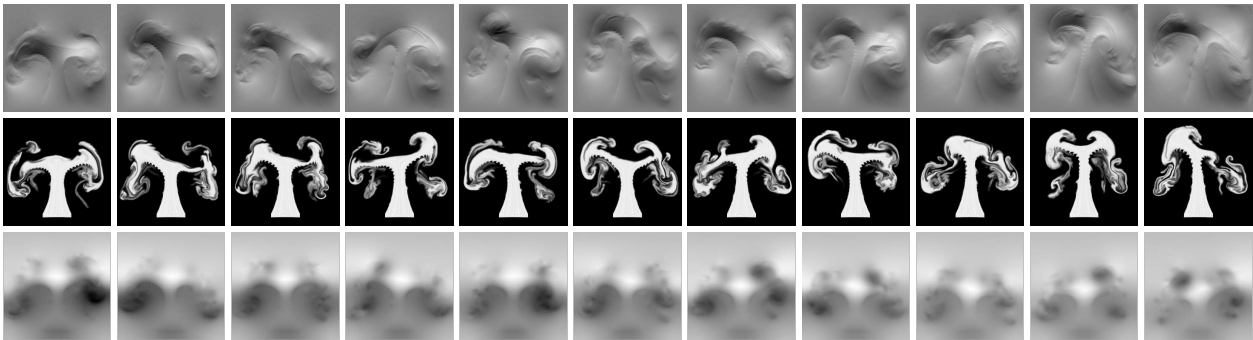
For the liquid data, a solver based on the fluid implicit particle (FLIP) method (Zhu & Bridson, 2005) was employed. It is a hybrid Eulerian-Lagrangian approach that replaces the Semi-Lagrangian advection scheme with particle based advection to reduce numerical dissipation. Still, this method is not optimal as we experienced problems such as mass loss, especially for larger noise values.

The simulation setup consists of a large breaking dam and several smaller liquid areas for more detailed splashes. After the dam hits the simulation boundary, a large, single drop of liquid is created in the middle of the domain that hits the already moving liquid surface. Then, the extrapolated level set values, binary indicator flags, and the velocity fields (separated by dimension) are saved. Some examples are shown in Fig. 10. The list of varied parameters include:

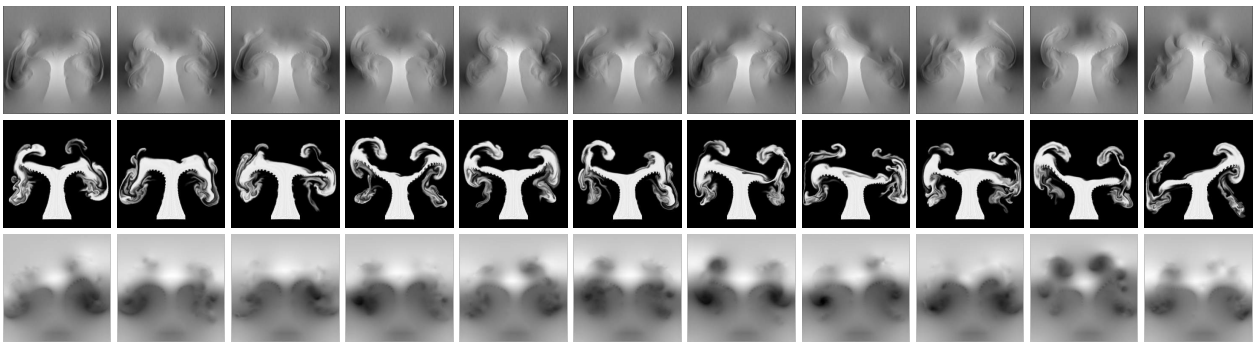
- Radius of the liquid drop
- Position of the drop in x- and y-direction
- Amount of additional gravity force in x- and y-direction
- Strength of noise added to the velocity field



(a) Data samples generated without noise: tiny output changes following generation ordering

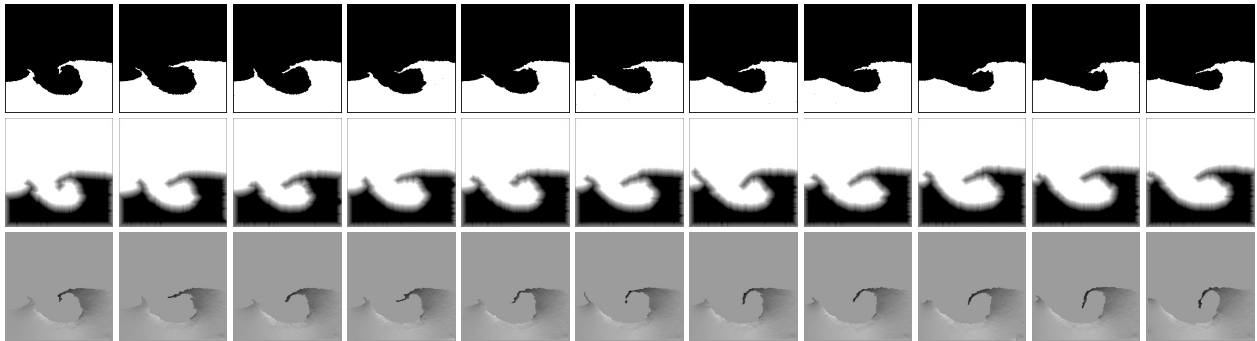


(b) Normal training data samples with noise: larger output changes but ordering still applies

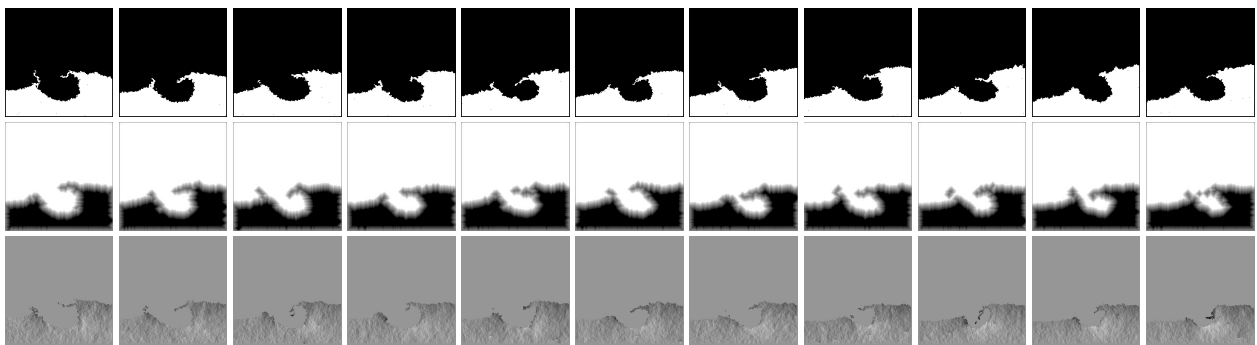


(c) Outlier data samples: noise can override the generation ordering by chance

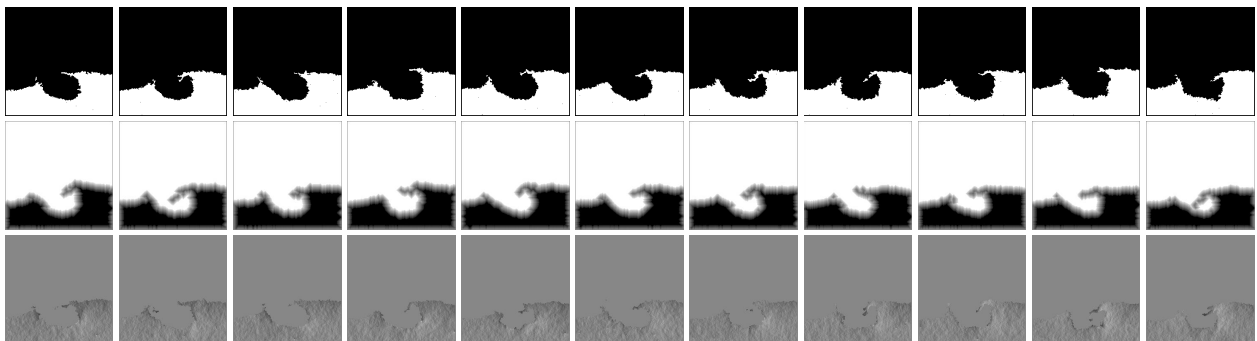
Figure 9. Various smoke simulation examples using one component of the velocity (top rows), the density (middle rows), and the pressure field (bottom rows).



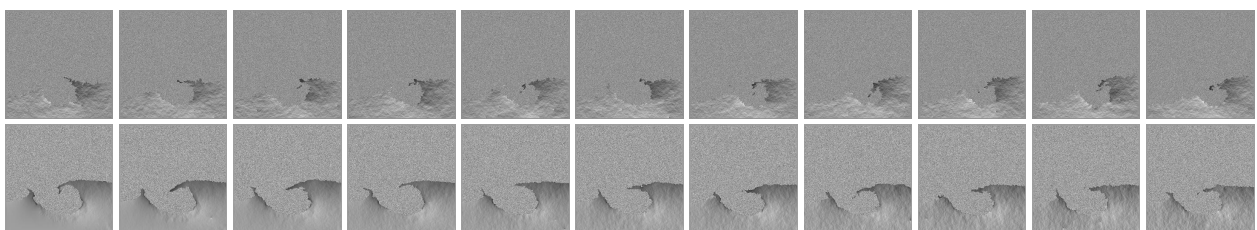
(a) Data samples generated without noise: tiny output changes following generation ordering



(b) Normal training data samples with noise: larger output changes but ordering still applies

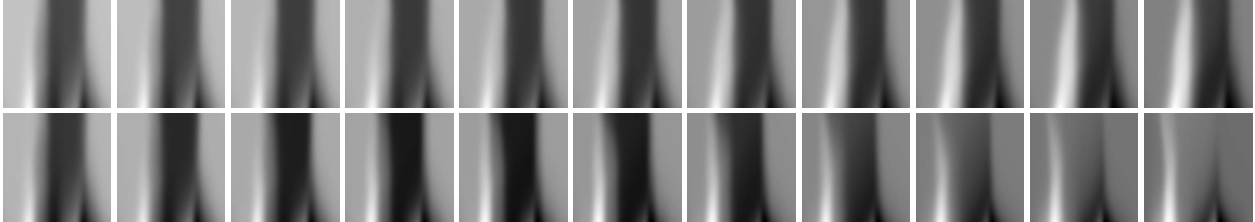


(c) Outlier data samples: noise can override the generation ordering by chance

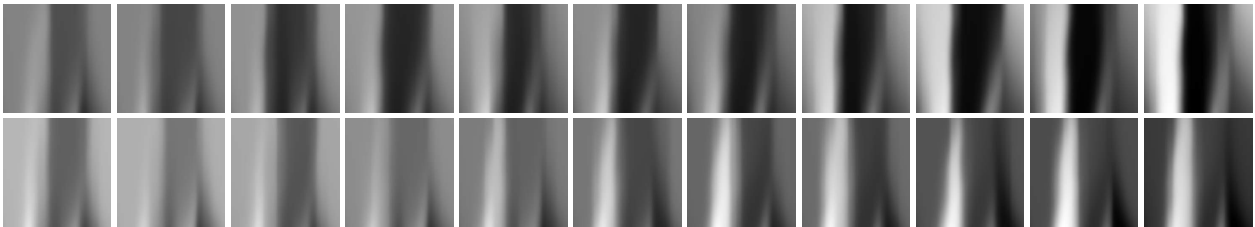


(d) Data samples from test set with additional background noise

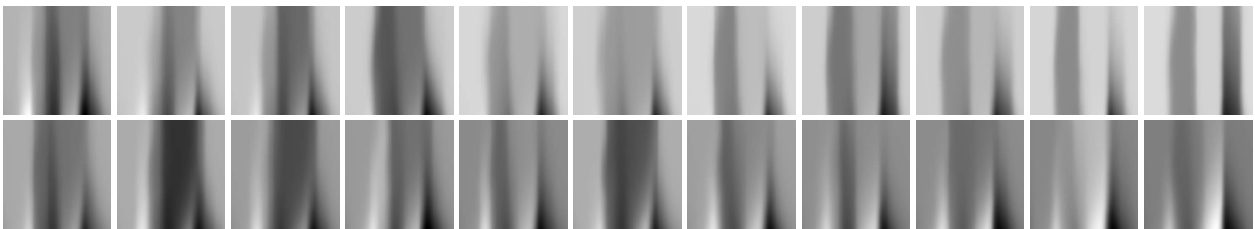
Figure 10. Several liquid simulation examples using the binary indicator flags (top rows), the extrapolated level set values (middle rows), and one component of the velocity field (bottom rows) for the training data and only the velocity field for the test data.



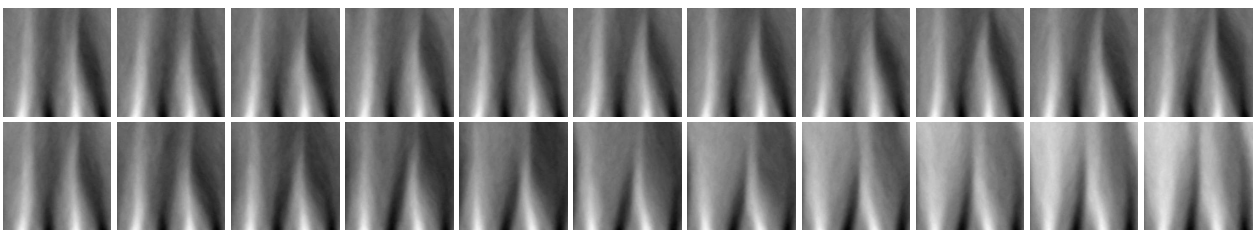
(a) Data samples generated without noise: tiny output changes following generation ordering



(b) Normal training data samples with noise: larger output changes but ordering still applies

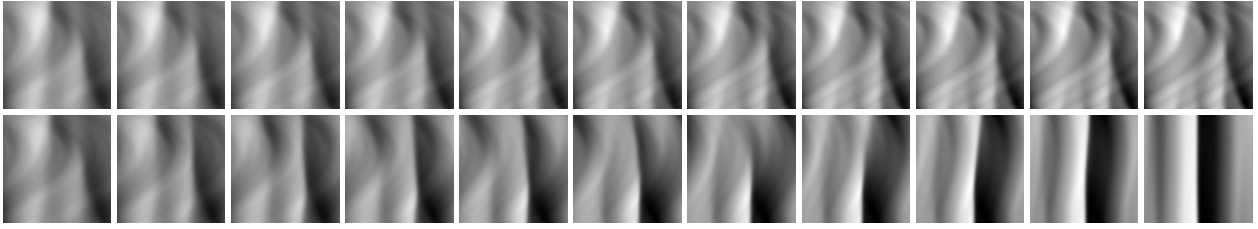


(c) Outlier data samples: noise can override the generation ordering by chance

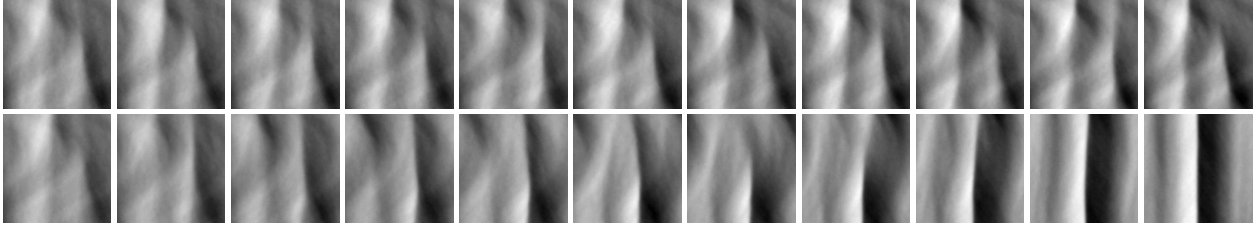


(d) Data samples from test set with additional background noise

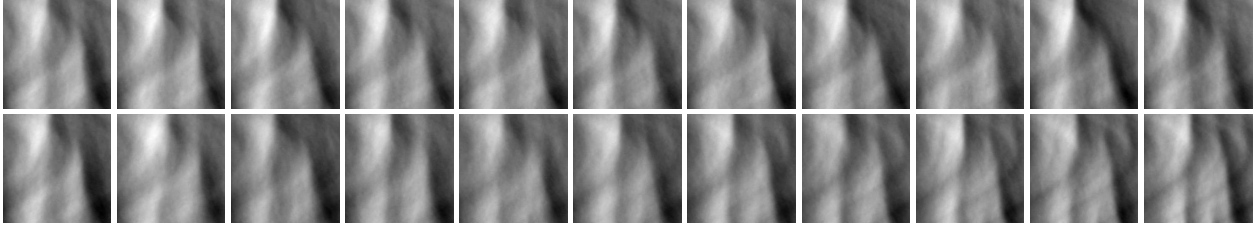
Figure 11. Various examples from the Advection-Diffusion equation using the density field.



(a) Data samples generated without noise: tiny output changes following generation ordering



(b) Normal training data samples with noise: larger output changes but ordering still applies



(c) Outlier data samples: noise can override the generation ordering by chance

Figure 12. Different simulation examples from the Burger’s equation using the velocity field.

The liquid training set consists of 792 sequences and the validation set of 198 sequences with different random seeds. For the liquid test set, additional background noise was added to the velocity field of the simulations as displayed in Fig. 10(d). Because this only alters the velocity field, the extrapolated level set values and binary indicator flags are not used for this data set, leading to 132 sequences.

D.2. Advection-Diffusion and Burger’s Equation

For these PDEs, our solvers only discretize and solve the corresponding equation in 1D. Afterwards, the different time steps of the solution process are concatenated along a new dimension to form 2D data with one spatial and one time dimension.

ADVECTION-DIFFUSION EQUATION

This equation describes how a passive quantity is transported inside a velocity field due to the processes of advection and diffusion. Eq. (7) is the simplified Advection-Diffusion equation with constant diffusivity and no sources or sinks.

$$\frac{\partial d}{\partial t} = \nu \nabla^2 d - u \cdot \nabla d, \quad (7)$$

where d denotes the density, u is the velocity, and ν is the kinematic viscosity (also known as diffusion coefficient) that determines the strength of the diffusion. Our solver employed a simple implicit time integration and a diffusion solver based on conjugate gradient without preconditioning. The initialization for the 1D fields of the simulations was created by overlaying multiple parameterized sine curves with random frequencies and magnitudes.

In addition, continuous forcing controlled by further parameterized sine curves was included in the simulations over time. In this case, the only initial conditions to vary are the forcing and initialization parameters of the sine curves and the strength of the added noise. From this PDE, only the passive density field was used as shown in Fig. 11. Overall, 798 sequences are included in the training set and 190 sequences with a different random initialization in the validation set.

For the Advection-Diffusion test set, the noise was instead added directly to the passive density field of the simulations. This results in 190 sequences with more small scale details as shown in Fig. 11(d).

BURGER’S EQUATION

This equation is very similar to the Advection-Diffusion equation and describes how the velocity field itself changes due to diffusion and advection:

$$\frac{\partial u}{\partial t} = \nu \nabla^2 u - u \cdot \nabla u. \tag{8}$$

Eq. (8) is known as the viscous form of the Burger’s equation that can develop shock waves, and again u is the velocity and ν denotes the kinematic viscosity. Our solver for this PDE used a slightly different implicit time integration scheme, but the same diffusion solver as used for the Advection-Diffusion equation.

The simulation setup and parameters were also the same; the only difference is that the velocity field instead of the density is exported. As a consequence, the data in Fig. 12 looks relatively similar to those from the Advection-Diffusion equation. The training set features 782 sequences, and the validation set contains 204 sequences with different random seeds.

D.3. Other Data-Sets

The remaining data sets are not based on PDEs and thus not generated with the proposed method. The data is only used to test the generalization of the discussed metrics and not for training or validation. The Shapes test set contains 160 sequences, the Video test set consists 131 sequences, and the TID test set features 216 sequences.

SHAPES

This data set tests if the metrics are able to track simple, moving geometric shapes. To create it, a straight path between two random points inside the domain is generated

and a random shape is moved along this path in steps of equal distance. The size of the used shape depends on the distance between the start and end point such that a significant fraction of the shape overlaps between two consecutive steps. It is also ensured that no part of the shape leaves the domain at any step by using a sufficiently big boundary area when generating the path.

With this method, multiple random shapes for a single data sample are produced, and their paths can overlap such that they occlude each other to provide an additional challenge. All shapes are moved in their parametric representation, and only when exporting the data, they are discretized onto a fixed binary grid. To add more variations to this simple approach, we also apply them in a non-binary way with smoothed edges and include additive Gaussian noise over the entire domain. Examples are shown in Fig. 13.

VIDEO

For this data set, different publicly available video recordings were acquired and processed in three steps. First, videos with abrupt cuts, scene transitions, or camera movements were discarded, and afterwards the footage was broken down into single frames. Then, each frame was resized to match the spatial size of our other data by linear interpolation. Since directly using consecutive frames is no challenge for any analyzed metric and all of them recovered the ordering almost perfectly, we achieved a more meaningful data set by skipping several intermediate frames. For the final data set, we defined the first frame of every video as the reference and collected subsequent frames in an interval step of ten frames as the increasingly different variations. Some data examples can be found in Fig. 14.

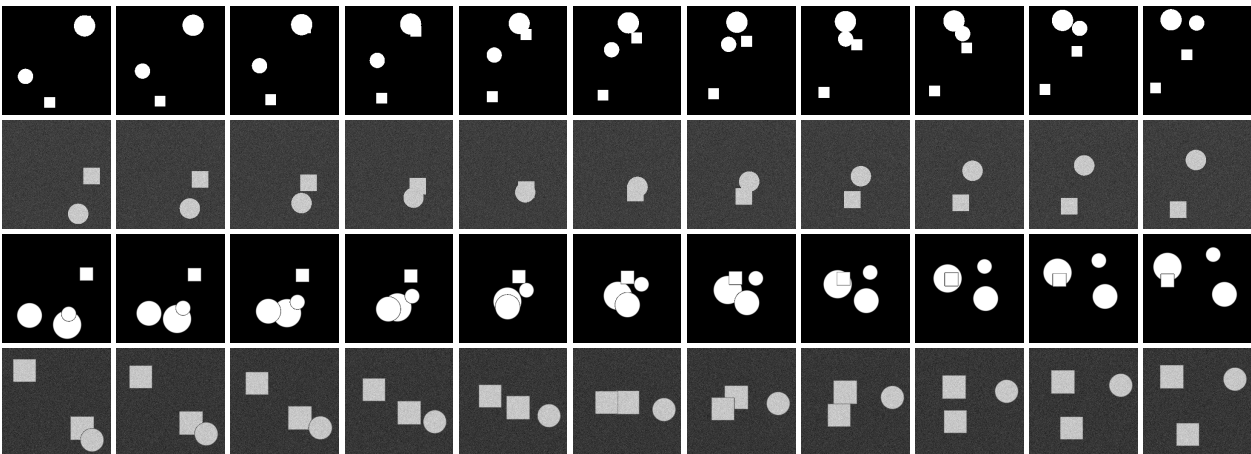


Figure 13. Examples from the shapes data set using a field with only binary shape values (first row), shape values with additional noise (second row), smoothed shape values (third row), and smoothed values with additional noise (fourth row).



Figure 14. Multiple examples from the video data set.

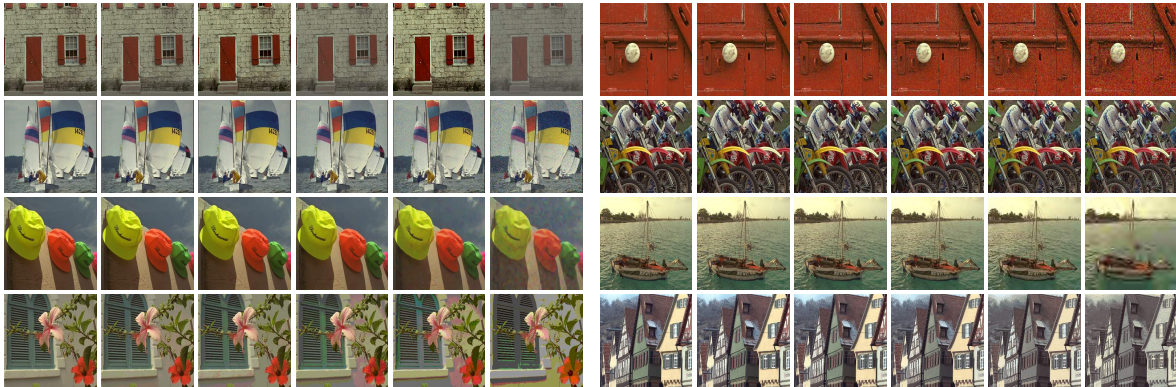


Figure 15. Examples from the TID2013 data set proposed by Ponomarenko et al.. Displayed are a change of contrast, three types of noise, denoising, jpg2000 compression, and two color quantizations (from left to right and top to bottom).

TID2013

This data set was created by Ponomarenko et al. and used without any further modifications. It consists of 25 reference images with 24 distortion types in five levels. As a result, it is not directly comparable to our data sets; thus, it is excluded from the test set aggregations. The distortions focus on various types of noise, image compression, and color changes. Fig. 15 contains examples from the data set.

D.4. Hardware

Data generation, training, and metric evaluations were performed on a machine with an Intel i7-6850 (3.60Ghz) CPU and an NVIDIA GeForce GTX 1080 Ti GPU.

E. Real-World Data

Below, we give details of the three data sets used for the evaluation in Section 6.3 of the main paper.

E.1. ScalarFlow

The *ScalarFlow* data set (Eckert et al., 2019) contains 3D velocities of real-world scalar transport flows reconstructed from multiple camera perspectives. For our evaluation, we cropped the volumetric $100 \times 178 \times 100$ grids to $100 \times 160 \times 100$ such that they only contain the area of interest and convert them to 2D with two variants: either by using the center slice or by computing the mean along the z-dimension. Afterwards, the velocity vectors are split by channels, linearly interpolated to 256×256 , and then normalized. Variations for each reconstructed plume are acquired by using frames in equal temporal intervals. We employed the velocity field reconstructions from 30 plumes (with simulation IDs 0 – 29) for both compression methods. Fig. 16 shows some example sequences.

E.2. Johns Hopkins Turbulence Database

The Johns Hopkins Turbulence Database (*JHTDB*) (Perlman et al., 2007) features various data sets of 3D turbu-

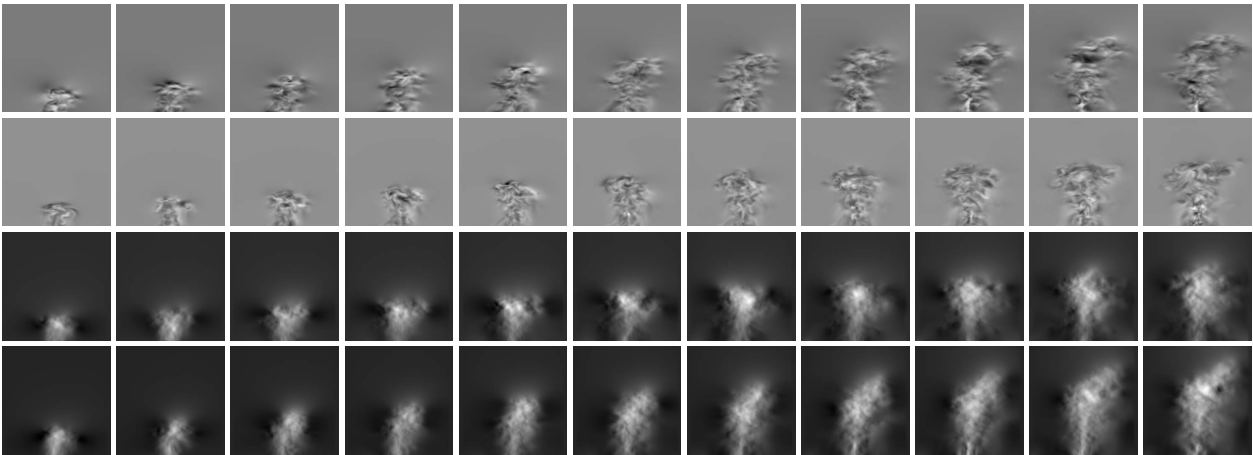


Figure 16. Four different smoke plume examples of the processed *ScalarFlow* data set using one of the three velocity components. The two top rows show the center slice, and the two bottom rows show the mean along the z-dimension. The temporal interval between each image is ten simulation time steps.

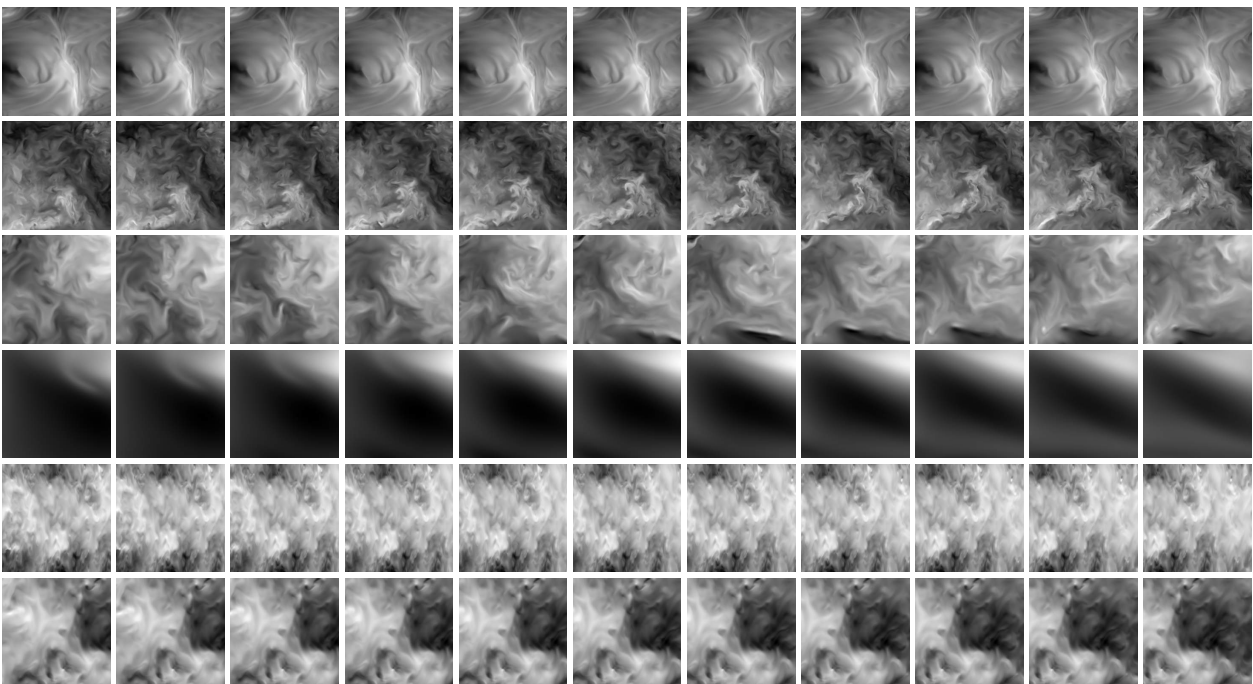


Figure 17. Data samples extracted from the Johns Hopkins Turbulence Database with a spatial or temporal interval of ten using one of the three velocity components. From top to bottom: *mhd1024* and *isotropic1024coarse* (varied time step), *isotropic4096* and *rotstrat4096* (varied z-position), *channel* and *channel5200* (varied x-position).

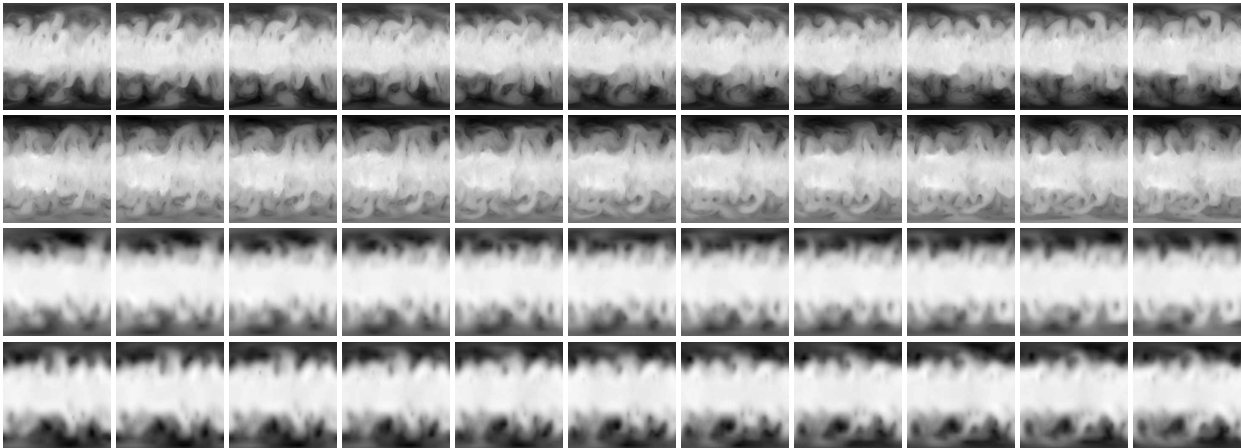


Figure 18. Examples of the processed *WeatherBench* data: high-res temperature data *1.40625deg/temperature* (upper two rows) and low-res geopotential data *5.625deg/geopotential_500* (lower two rows). The temporal interval spacing between the images is twenty hours.

lent flow fields created with direct numerical simulations (DNS). Here, we used three forced isotropic turbulence data sets with different resolutions (*isotropic1024coarse*, *isotropic1024fine*, and *isotropic4096*), two channel flows with different Reynolds numbers (*channel* and *channel-5200*), the forced magneto-hydrodynamic isotropic turbulence data set (*mhd1024*), and the rotating stratified turbulence data set (*rotstrat4096*).

For the evaluation, five 256×256 reference slices in the x/y -plane from each of the seven data sets are used. The spatial and temporal position of each slice is randomized within the bounds of the corresponding simulation domain. We normalize the value range and split the velocity vectors by component for an individual evaluation. Variants for each reference are created by gradually varying the x - and z -position of the slice in equal intervals. The temporal position of each slice is varied as well if a sufficient amount of temporally resolved data is available (for *isotropic1024coarse*, *isotropic1024fine*, *channel*, and *mhd1024*). This leads to 216 sequences in total. Fig. 17 shows examples from six of the *JHTDB* data sets.

E.3. WeatherBench

The *WeatherBench* repository (Rasp et al., 2020) represents a collection of various weather measurements of different atmospheric quantities such as precipitation, cloud coverage, wind velocities, geopotential, and temperature. The data ranges from 1979 to 2018 with a fine temporal resolution and is stored on a Cartesian latitude-longitude grid of the earth. In certain subsets of the data, an additional dimension such as altitude or pressure levels is available. As all measurements are available as scalar fields, only a linear interpolation to the correct input size and a normalization was necessary in order to prepare the data. We used the low-

resolution geopotential data set at 500hPa (i.e., at around 5.5km height) with a size of 32×64 yielding smoothly changing features when upsampling the data. In addition, the high-res temperature data with a size of 128×256 for small scale details was used. For the temperature field, we used the middle atmospheric pressure level at 850hPa corresponding to an altitude of 1.5km in our experiments.

To create sequences with variations for a single time step of the weather data, we used frames in equal time intervals, similar to the *ScalarFlow* data. Due to the very fine temporal discretization of the data, we only use a temporal interval of two hours as the smallest interval step of one in Fig. 19. We sampled three random starting points in time from each of the 40 years of measurements, resulting in 120 temperature and geopotential sequences overall. Fig. 18 shows a collection of example sequences.

E.4. Detailed Results

For each of the variants explained in the previous sections, we create test sets with six different spatial and temporal intervals. Fig. 19 shows the combined Spearman correlation of the sequences for different interval spacings when evaluating various metrics. For the results in Fig. 7 in the main paper, all correlation values shown here are aggregated by data source via mean and standard deviation.

While our metric reliably recovers the increasing distances within the data sets, the individual measurements exhibit interesting differences in terms of their behavior for varying distances. As *JHTDB* and *WeatherBench* contain relatively uniform phenomena, a larger step interval creates more difficult data as the simulated and measured states contain changes that are more and more difficult to analyze along a sequence. For *ScalarFlow*, on the other hand, the diffi-

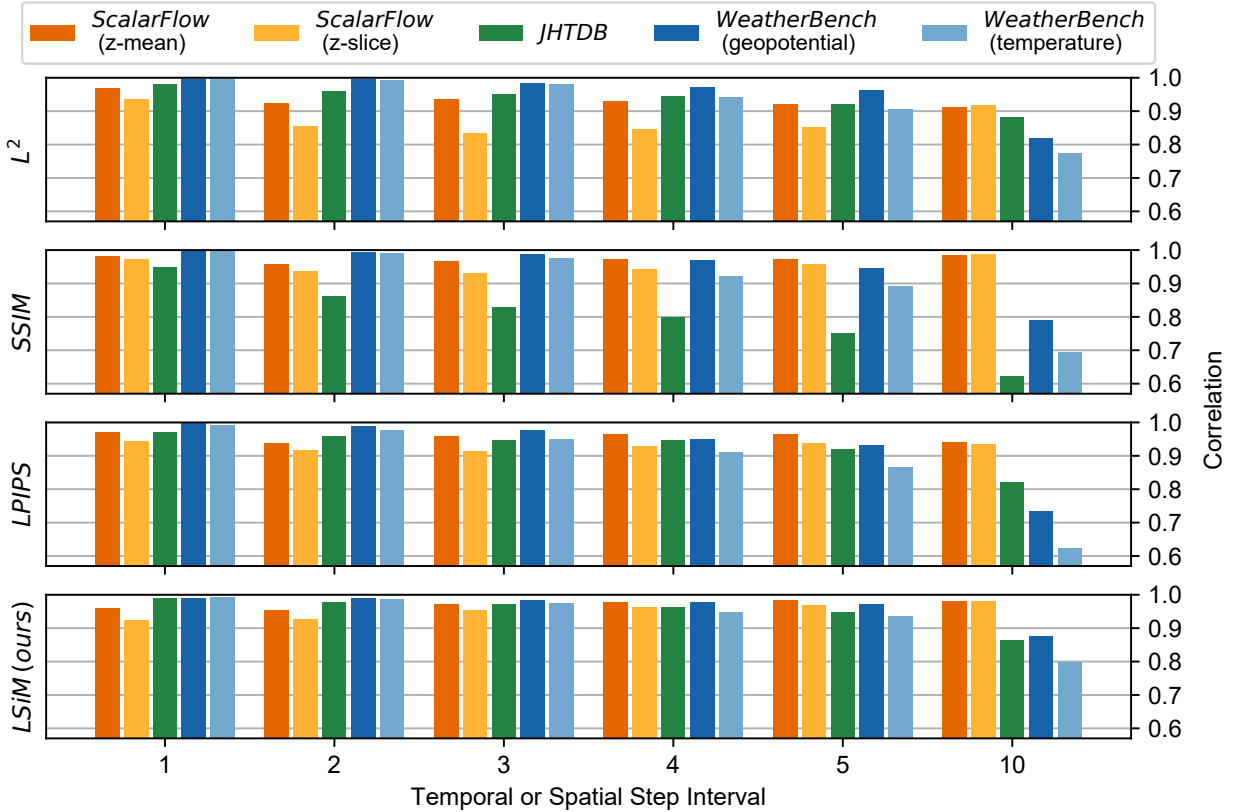


Figure 19. Detailed breakdown of the results when evaluating $LSiM$ on the individual data sets of *ScalarFlow* (30 sequences each), *JHTDB* (90 sequences each), and *WeatherBench* (120 sequences each) with different step intervals.

culty decreases for larger intervals due to the large-scale motion of the reconstructed plumes. As a result of buoyancy forces, the observed smoke rises upwards into areas where no smoke has been before. For the network, this makes predictions relatively easy as the large-scale translations are indicative of the temporal progression, and small scale turbulence effects can be largely ignored. For this data set, smaller intervals are more difficult as the overall shape of the plume barely changes while the complex evolution of small scale features becomes more important.

Overall, the $LSiM$ metric recovers the ground truth ordering of the sequences very well as indicated by the consistently high correlation values in Fig. 19. The other metrics come close to these results on certain sub-datasets but are significantly less consistent. $SSiM$ struggles on *JHTDB* across all interval sizes, and $LPIPS$ cannot keep up on *WeatherBench*, especially for larger intervals. L^2 is more stable overall, but consistently stays below the correlation achieved by $LSiM$.

F. Additional Evaluations

In the following, we demonstrate other ways to compare the performance of the analyzed metrics on our data sets. In

Tab. 1, the Pearson correlation coefficient is used instead of Spearman’s rank correlation coefficient. While Spearman’s correlation measures monotonic relationships by using ranking variables, it directly measures linear relationships.

The results in Tab. 1 match very closely to the values computed with Spearman’s rank correlation coefficient. The best performing metrics in both tables are identical; only the numbers slightly vary. Since a linear and a monotonic relation describes the results of the metrics similarly well, there are no apparent non-linear dependencies that cannot be captured using the Pearson correlation.

In the Tables 2 and 3, we employ a different, more intuitive approach to determine combined correlation values for each data set using the Pearson correlation. We are no longer analyzing the entire predicted distance distribution and the ground truth distribution at once as done above. Instead, we individually compute the correlation between the ground truth and the predicted distances for the single data samples of the data set. From the single correlation values, we compute the mean and standard deviations shown in the tables. Note that this approach potentially produces less accurate comparison results, as small errors in the individual

Learning Similarity Metrics for Numerical Simulations

Table 1. Performance comparison on validation and test data sets measured in terms of the Pearson correlation coefficient of ground truth against predicted distances. **Bold+underlined** values show the best performing metric for each data set, **bold** values are within a 0.01 error margin of the best performing, and *italic* values are 0.2 or more below the best performing. On the right a visualization of the combined test data results is shown for selected models.

Metric	Validation data sets				Test data sets					
	Smo	Liq	Adv	Bur	TID	LiqN	AdvD	Sha	Vid	All
L^2	0.66	0.80	0.72	0.60	0.82	0.73	0.55	<i>0.66</i>	0.79	0.60
SSIM	0.69	0.74	0.76	0.70	0.78	<i>0.26</i>	0.69	<i>0.49</i>	0.73	0.53
LPIPS v0.1.	0.63	0.68	0.66	0.71	0.85	<i>0.49</i>	0.61	0.84	0.83	0.65
<i>AlexNet</i> _{random}	0.63	0.69	0.67	0.65	0.83	0.64	0.63	0.74	0.81	0.65
<i>AlexNet</i> _{frozen}	0.66	0.69	0.68	0.71	0.85	<i>0.39</i>	0.61	0.86	0.83	0.64
Optical flow	0.63	<i>0.56</i>	<i>0.37</i>	<i>0.39</i>	<i>0.49</i>	<i>0.45</i>	<i>0.28</i>	<i>0.61</i>	0.74	<i>0.48</i>
Non-Siamese	0.77	0.84	0.78	0.74	0.67	0.81	0.64	<i>0.27</i>	0.79	0.60
<i>Skip</i> _{from scratch}	0.79	0.83	0.80	0.73	0.85	0.78	0.61	0.79	0.84	0.71
LSiM _{noiseless}	0.77	0.77	0.76	0.72	0.86	0.62	0.58	0.84	0.83	0.68
LSiM _{strong noise}	0.65	<i>0.64</i>	0.66	0.68	0.81	<i>0.39</i>	0.53	0.90	0.82	0.64
LSiM (ours)	0.78	0.82	0.79	0.74	0.86	0.79	0.58	0.87	0.82	0.72

Table 2. Performance comparison on validation data sets measured by computing mean and standard deviation (in brackets) of Pearson correlation coefficients (ground truth against predicted distances) from individual data samples. **Bold+underlined** values show the best performing metric for each data set, **bold** values are within a 0.01 error margin of the best performing, and *italic* values are 0.2 or more below the best performing. On the right a visualization of the combined test data results is shown for selected models.

Metric	Validation data sets			
	Smo	Liq	Adv	Bur
L^2	0.68 (0.27)	0.82 (0.18)	0.74 (0.24)	0.63 (0.33)
SSIM	0.71 (0.23)	0.75 (0.23)	0.79 (0.21)	0.73 (0.33)
LPIPS v0.1.	0.66 (0.29)	0.71 (0.24)	0.70 (0.29)	0.75 (0.28)
<i>AlexNet</i> _{random}	0.65 (0.28)	0.71 (0.29)	0.71 (0.27)	0.68 (0.31)
<i>AlexNet</i> _{frozen}	0.69 (0.27)	0.72 (0.25)	0.71 (0.27)	0.74 (0.29)
Optical flow	0.66 (0.38)	<i>0.59 (0.47)</i>	<i>0.38 (0.52)</i>	<i>0.41 (0.49)</i>
Non-Siamese	0.80 (0.19)	0.87 (0.14)	0.81 (0.20)	0.76 (0.32)
<i>Skip</i> _{from scratch}	0.81 (0.19)	0.85 (0.16)	0.82 (0.19)	0.77 (0.30)
LSiM _{noiseless}	0.79 (0.21)	0.79 (0.20)	0.79 (0.23)	0.76 (0.29)
LSiM _{strong noise}	0.67 (0.28)	<i>0.66 (0.29)</i>	0.68 (0.30)	0.70 (0.32)
LSiM (ours)	0.81 (0.20)	0.84 (0.16)	0.81 (0.19)	0.78 (0.28)

Table 3. Performance comparison on test data sets measured by computing mean and std. dev. (in brackets) of Pearson correlation coefficients (ground truth against predicted distances) from individual data samples. **Bold+underlined** values show the best performing metric for each data set, **bold** values are within a 0.01 error margin of the best performing, and *italic* values are 0.2 or more below the best performing.

Metric	Test data sets					
	TID	LiqN	AdvD	Sha	Vid	All
L^2	0.84 (0.08)	0.75 (0.18)	0.57 (0.38)	<i>0.67 (0.18)</i>	0.84 (0.27)	0.69 (0.29)
SSIM	0.81 (0.20)	<i>0.26 (0.38)</i>	0.71 (0.31)	<i>0.53 (0.32)</i>	0.77 (0.28)	0.58 (0.38)
LPIPS v0.1.	0.87 (0.11)	<i>0.51 (0.34)</i>	0.63 (0.34)	0.85 (0.14)	0.87 (0.22)	0.71 (0.31)
<i>AlexNet</i> _{random}	0.84 (0.10)	0.67 (0.24)	0.65 (0.33)	0.74 (0.18)	0.85 (0.26)	0.72 (0.28)
<i>AlexNet</i> _{frozen}	0.86 (0.11)	<i>0.41 (0.37)</i>	0.64 (0.34)	0.87 (0.14)	0.87 (0.22)	0.70 (0.34)
Optical flow	0.74 (0.67)	<i>0.50 (0.34)</i>	<i>0.32 (0.53)</i>	<i>0.63 (0.45)</i>	0.78 (0.45)	<i>0.53 (0.49)</i>
Non-Siamese	0.87 (0.12)	0.84 (0.12)	0.66 (0.34)	<i>0.31 (0.45)</i>	0.83 (0.26)	0.64 (0.39)
<i>Skip</i> _{from scratch}	0.87 (0.12)	0.80 (0.16)	0.63 (0.37)	0.80 (0.17)	0.87 (0.20)	0.76 (0.27)
LSiM _{noiseless}	0.87 (0.11)	<i>0.64 (0.29)</i>	0.60 (0.38)	0.86 (0.15)	0.86 (0.22)	0.73 (0.31)
LSiM _{strong noise}	0.83 (0.12)	<i>0.39 (0.38)</i>	0.55 (0.36)	0.91 (0.17)	0.86 (0.25)	0.67 (0.37)
LSiM (ours)	0.88 (0.10)	0.81 (0.15)	0.60 (0.37)	0.88 (0.16)	0.85 (0.23)	0.77 (0.28)

computations can accumulate to larger deviations in mean and standard deviation. Still, both tables lead to very similar conclusions: The best performing metrics are almost the same, and low combined correlation values match with results that have a high standard deviation and a low mean.

Fig. 20 shows a visualization of predicted distances c against ground truth distances d for different metrics on every sample from the test sets. Each plot contains over 6700 individual data points to illustrate the global distance distributions created by the metrics, without focusing on single cases. A theoretical optimal metric would recover a perfectly narrow distribution along the line $c = d$, while worse metrics recover broader, more curved distributions. Overall, the sample distribution of an L^2 metric is very wide. $LPIPS$ manages to follow the optimal diagonal a lot better, but our approach approximates it with the smallest deviations, as also shown in the tables above. The L^2 metric performs very poorly on the shape data indicated by the too steeply increasing blue lines that flatten after a ground truth distance of 0.3. $LPIPS$ already significantly reduces this problem, but $LSiM$ still works slightly better.

A similar issue is visible for the Advection-Diffusion data, where for L^2 a larger number of red samples is below the

optimal $c = d$ line, than for the other metrics. $LPIPS$ has the worst overall performance for liquid test set, indicated by the large number of fairly chaotic green lines in the plot. On the video data, all three metrics perform similarly well.

A fine-grained distance evaluation in 200 steps of L^2 and our $LSiM$ metric via the mean and standard deviation of different data samples is shown in Fig. 21. Similar to Fig. 20, the mean of an optimal metric would follow the ground truth line with a standard deviation of zero, while the mean of worse metrics deviates around the line with a high standard deviation. The plot on the left combines eight samples with different seeds from the `Sha` data set, where only a single shape is used. Similarly, the center plot aggregates eight samples from `Sha` with more than one shape. The right plot shows six data samples from the `LiqN` test set that vary by the amount of noise that was injected into the simulation.

The task of only tracking a single shape in the example on the left is the easiest of the three shown cases. Both metrics have no problem to recover the position change until a variation of 0.4, where L^2 can no longer distinguish between the different samples. Our metric recovers distances with a continuously rising mean and a very low standard deviation. The task in the middle is already harder, as multiple shapes

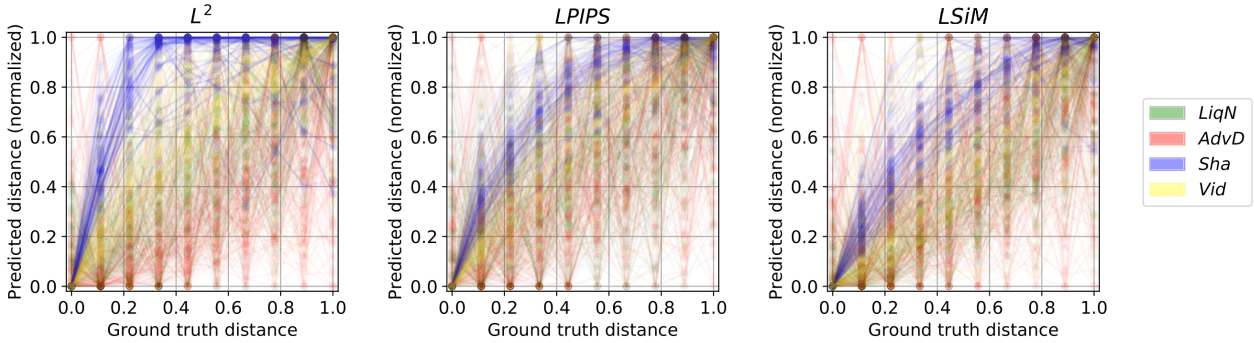


Figure 20. Distribution evaluation of ground truth distances against normalized predicted distances for L^2 , $LPIPS$ and $LSiM$ on all test data (color coded).

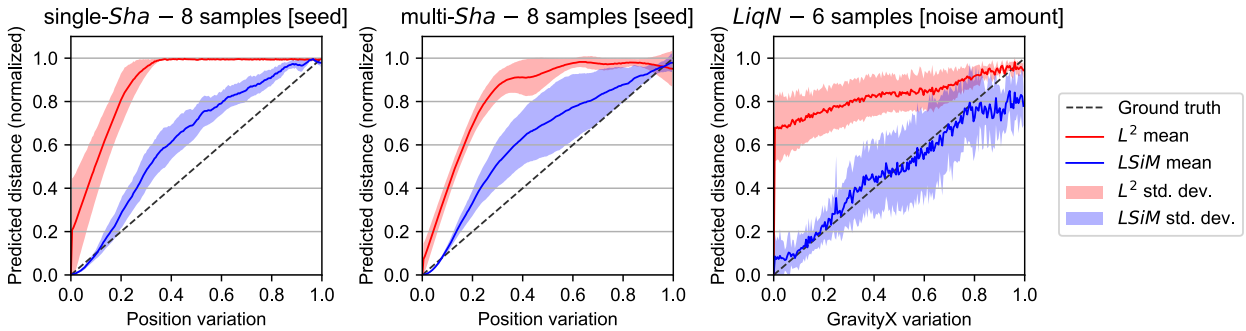


Figure 21. Mean and standard deviation of normalized distances over multiple data samples for L^2 and $LSiM$. The samples differ by the quantity displayed in brackets. Each data sample uses 200 parameter variation steps instead of 10 like the others in our data sets. For the shape data the position of the shape varies and for the liquid data the gravity in x-direction is adjusted.

can occlude each other during the position changes. Starting at a position variation of 0.4, both metrics have a quite high standard deviation, but the proposed method stays closer to the ground truth line. L^2 shows a similar issue as before because it flattens relatively fast. The plot on the right features the hardest task. Here, both metrics perform similar as each has a different problem in addition to an unstable mean. Our metric stays close to the ground truth, but has a quite high standard deviation starting at about a variation of 0.4. The standard deviation of L^2 is lower, but instead it starts off with a big jump from the first few data points. To some degree, this is caused by the normalization of the plots, but it still overestimates the relative distances for small variations in the simulation parameter.

These findings also match with the distance distribution evaluations in Fig. 20 and the tables above: Our method has a significant advantage over shallow metrics on shape data, while the differences of both metrics become much smaller for the liquid test set.

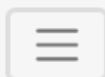
G. Notation

In this work, we follow the notation suggested by Goodfellow et al.. Vector quantities are displayed in bold, and tensors use a sans-serif font. Double-barred letters indicate sets or vector spaces. The following symbols are used:

\mathbb{R}	Real numbers	o_0, o_1, \dots, o_n	Series of outputs of a simulation with increasing ground truth distance to o_0
i, j	Indexing in different contexts	Δ	Amount of change in a single simulation parameter
\mathbb{I}	Input space of the metric, i.e., color images/field data of size $224 \times 224 \times 3$	t_1, t_2, \dots, t_t	Time steps of a numerical simulation
a	Dimension of the input space \mathbb{I} when flattened to a single vector	v	Variance of the noise added to a simulation
$\mathbf{x}, \mathbf{y}, \mathbf{z}$	Elements in the input space \mathbb{I}	\mathbf{c}	Ground truth distance distribution, determined by the data generation via Δ
\mathbb{L}	Latent space of the metric, i.e., sets of 3 rd order feature map tensors	\mathbf{d}	Predicted distance distribution (supposed to match the corresponding \mathbf{c})
b	Dimension of the latent space \mathbb{L} when flattened to a single vector	$\bar{\mathbf{c}}, \bar{\mathbf{d}}$	Mean of the distributions \mathbf{c} and \mathbf{d}
$\tilde{\mathbf{x}}, \tilde{\mathbf{y}}, \tilde{\mathbf{z}}$	Elements in the latent space \mathbb{L} , corresponding to $\mathbf{x}, \mathbf{y}, \mathbf{z}$	$\ \dots\ _2$	Euclidean norm of a vector
w	Weights for the learned average aggregation (1 per feature map)	$m(\mathbf{x}, \mathbf{y})$	Entire function computed by our metric
p_0, p_1, \dots	Initial conditions / parameters of a numerical simulation	$m_1(\mathbf{x}, \mathbf{y})$	First part of $m(\mathbf{x}, \mathbf{y})$, i.e., base network and feature map normalization
n	Number of variations of a simulation parameter, thus determines length of the network input sequence	$m_2(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$	Second part of $m(\mathbf{x}, \mathbf{y})$, i.e., latent space difference and the aggregations
		G	3 rd order feature tensor from one layer of the base network
		g_b, g_c, g_x, g_y	Batch (g_b), channel (g_c), and spatial dimensions (g_x, g_y) of G
		f	Optical flow network
		$f^{\mathbf{x}\mathbf{y}}, f^{\mathbf{y}\mathbf{x}}$	Flow fields computed by an optical flow network f from two inputs in \mathbb{I}
		$f_1^{\mathbf{x}\mathbf{y}}, f_2^{\mathbf{x}\mathbf{y}}$	Components of the flow field $f^{\mathbf{x}\mathbf{y}}$
		∇, ∇^2	Gradient (∇) and Laplace operator (∇^2)
		∂	Partial derivative operator
		t	Time in our PDEs
		u	Velocity in our PDEs
		ν	Kinematic viscosity / diffusion coefficient in our PDEs
		d, ρ	Density in our PDEs
		P	Pressure in the Navier-Stokes Equations
		g	Gravity in the Navier-Stokes Equations

References

- Eckert, M.-L., Um, K., and Thuerey, N. Scalarflow: A large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. *ACM Transactions on Graphics*, 38(6), 2019. doi:10.1145/3355089.3356545.
- Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016. URL <http://www.deeplearningbook.org>.
- He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi:10.1109/CVPR.2016.90.
- Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269, 2017. doi:10.1109/CVPR.2017.243.
- Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. FlowNet 2.0: Evolution of optical flow estimation with deep networks. *CoRR*, abs/1612.01925, 2016. URL <http://arxiv.org/abs/1612.01925>.
- Perlman, E., Burns, R., Li, Y., and Meneveau, C. Data exploration of turbulence simulations using a database cluster. In *SC '07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, pp. 1–11, 2007. doi:10.1145/1362622.1362654.
- Ponomarenko, N., Jin, L., Ieremeiev, O., Lukin, V., Egiazarian, K., Astola, J., Vozel, B., Chehdi, K., Carli, M., Battisti, F., and Kuo, C. C. J. Image database TID2013: Peculiarities, results and perspectives. *Signal Processing-Image Communication*, 30:57–77, 2015. doi:10.1016/j.image.2014.10.009.
- Rasp, S., Dueben, P., Scher, S., Weyn, J., Mouatadid, S., and Thuerey, N. Weatherbench: A benchmark dataset for data-driven weather forecasting. *CoRR*, abs/2002.00469, 2020. URL <http://arxiv.org/abs/2002.00469>.
- Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL <http://arxiv.org/abs/1505.04597>.
- Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018. doi:10.1109/CVPR.2018.00068.
- Zhu, Y. and Bridson, R. Animating sand as a fluid. In *ACM SIGGRAPH 2005 Papers*, pp. 965–972, New York, NY, USA, 2005. doi:10.1145/1186822.1073298.



Who holds the Copyright on a ICML paper

According to U.S. Copyright Office's page [What is a Copyright](#). When you create an original work you are the author and the owner and hold the copyright, unless you have an agreement to transfer the copyright to a third party such as the company or school you work for.

Authors do not transfer the copyright of their paper to ICML, instead they grant ICML a non-exclusive, perpetual, royalty-free, fully-paid, fully-assignable license to copy, distribute and publicly display all or part of the paper



The ICML Logo above may be used on presentations. Right-click and choose download. It is a vector graphic and may be used at any scale.


USEFUL LINKS

[Code of Conduct](#)

[Registration Cancellation Policy](#)

[About ICML](#)

CONTACT

 1269 Law Street, San Diego CA 92109

 [Email](#)

[ICML Proceedings at PMLR](#)

Learning Similarity Metrics for Volumetric Simulations with Multiscale CNNs

Georg Kohl, Li-Wei Chen, Nils Thuerey

Technical University of Munich
{georg.kohl, liwei.chen, nils.thuerey}@tum.de

Abstract

Simulations that produce three-dimensional data are ubiquitous in science, ranging from fluid flows to plasma physics. We propose a similarity model based on entropy, which allows for the creation of physically meaningful ground truth distances for the similarity assessment of scalar and vectorial data, produced from transport and motion-based simulations. Utilizing two data acquisition methods derived from this model, we create collections of fields from numerical PDE solvers and existing simulation data repositories. Furthermore, a multiscale CNN architecture that computes a volumetric similarity metric (*VolSiM*) is proposed. To the best of our knowledge this is the first learning method inherently designed to address the challenges arising for the similarity assessment of high-dimensional simulation data. Additionally, the tradeoff between a large batch size and an accurate correlation computation for correlation-based loss functions is investigated, and the metric’s invariance with respect to rotation and scale operations is analyzed. Finally, the robustness and generalization of *VolSiM* is evaluated on a large range of test data, as well as a particularly challenging turbulence case study, that is close to potential real-world applications.

1 Introduction

Making comparisons is a fundamental operation that is essential for any kind of computation. This is especially true for the simulation of physical phenomena, as we are often interested in comparing simulations against other types of models or measurements from a physical system. Mathematically, such comparisons require metric functions that determine scalar distance values as a similarity assessment. A fundamental problem is that traditional comparisons are typically based on simple, element-wise metrics like the L^1 or L^2 distances, due to their computational simplicity and a lack of alternatives. Such metrics can work reliably for systems with few elements of interest, e.g. if we want to analyze the position of a moving object at different points in time, matching our intuitive understanding of distances. However, more complex physical problems often exhibit large numbers of degrees of freedom, and strong dependencies between elements in their solutions. Those coherences should be considered when comparing physical data, but element-wise operations by definition ignore such interactions between elements. With the curse of

dimensionality, this situation becomes significantly worse for systems that are modeled with dense grid data, as the number of interactions grows exponentially with a linearly increasing number elements. Such data representations are widely used, e.g. for medical blood flow simulations (Olufsen et al. 2000), climate and weather predictions (Stocker et al. 2014), and even the famous unsolved problem of turbulence (Holmes et al. 2012). Another downside of element-wise metrics is that each element is weighted equally, which is typically suboptimal; e.g. smoke plumes behave differently along the vertical dimension due to gravity or buoyancy, and small key features like vortices are more indicative of the fluid’s general behavior than large areas of near constant flow (Pope 2000).

In the image domain, neural networks have been employed for similarity models that can consider larger structures, typically via training with class labels that provide semantics, or with data that encodes human perception. Similarly, physical systems exhibit spatial and temporal coherence due to the underlying laws of physics that can be utilized. In contrast to previous work on simulation data (Kohl, Um, and Thuerey 2020), we derive an entropy-based similarity model to robustly learn similarity assessments of scalar and vectorial volumetric data. Overall, our work contributes the following:

- We propose a novel similarity model based on the entropy of physical systems. It is employed to synthesize sequences of volumetric physical fields suitable for metric learning.
- We show that our Siamese multiscale feature network results in a stable metric that successfully generalizes to new physical phenomena. To the best of our knowledge this is the first learned metric inherently designed for the similarity assessment of volumetric fields.
- The metric is employed to analyze turbulence in a case study, and its invariance to rotation and scale are evaluated. In addition, we analyze correlation-based loss functions with respect to their tradeoff between batch size and accuracy of correlation computation.

The central application of the proposed *VolSiM* metric is the similarity assessment of new physical simulation methods, numerical or learning-based, against a known ground truth.¹

¹Our source code, datasets, and ready-to-use models are available at <https://github.com/tum-pbs/VOLSIM>. For a version of this work with an appendix also see <https://arxiv.org/abs/2202.04109>.

This ground truth can take the form of measurements, higher resolution simulations, or existing models. Similar to perceptual losses for computer vision tasks, the trained metric can also be used as a differentiable similarity loss for various physical problems. We refer to Thuerey et al. (2021) for an overview of such problems and different learning methods to approach them.

2 Related Work

Apart from simple L^n distances, the two metrics peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) from Wang et al. are commonly used across disciplines for the similarity assessment of data. Similar to the underlying L^2 distance, PSNR shares the issues of element-wise metrics (Huynh-Thu and Ghanbari 2008, 2012). SSIM computes a more intricate function, but it was shown to be closely related to PSNR (Horé and Ziou 2010) and thus has similar problems (Nilsson and Akenine-Möller 2020). Furthermore, statistical tools like the Pearson correlation coefficient PCC (Pearson 1920) and Spearman’s rank correlation coefficient SRCC (Spearman 1904) can be employed as a simple similarity measurement. There are several learning-based metrics specialized for different domains such as rendered (Andersson et al. 2020) and natural images (Bosse et al. 2016), interior object design (Bell and Bala 2015), audio (Avgoustinakis et al. 2020), and haptic signals (Kumari, Chaudhuri, and Chaudhuri 2019).

Especially for images, similarity measurements have been approached in various ways, but mostly by combining deep embeddings as perceptually more accurate metrics (Prashnani et al. 2018; Talebi and Milanfar 2018). These metrics can be employed for various applications such as image super-resolution (Johnson, Alahi, and Fei-Fei 2016) or generative tasks (Dosovitskiy and Brox 2016). Traditional metric learning for images typically works in one of two ways: Either, the training is directly supervised by learning from manually created labels, e.g. via two-alternative forced choice where humans pick the most similar option to a reference (Zhang et al. 2018), or the training is indirectly semi-supervised through images with class labels and a contrastive loss (Chopra, Hadsell, and LeCun 2005; Hadsell, Chopra, and LeCun 2006). In that case, triplets of reference, same class image, and other class images are sampled, and the corresponding latent space representations are pulled together or pushed apart. We refer to Roth et al. (2020) for an overview of different training strategies for learned image metrics. In addition, we study the behavior of invariance and equivariance to different transformations, which was targeted previously for rotational symmetries (Weiler et al. 2018; Chidester et al. 2019) and improved generalization (Wang, Walters, and Yu 2021).

Similarity metrics for simulation data have not been studied extensively yet. Siamese networks for finding similar fluid descriptors have been applied to smoke flow synthesis, where a highly accurate similarity assessment is not necessary (Chu and Thuerey 2017). Um et al. (2017; 2021) used crowd-sourced user studies for the similarity assessment of liquid simulations which rely on relatively slow and expensive human evaluations. Scalar 2D simulation data was previously compared with a learned metric using a Siamese network

(Kohl, Um, and Thuerey 2020), but we overcome methodical weaknesses and improve upon the performance of their work. Their *LSiM* method relies on a basic feature extractor based on common classification CNNs, does not account for the long-term behavior of different systems with respect to entropy via a similarity model during training, and employs a simple heuristic to generate suitable data sequences.

3 Modeling Similarity of Simulations

To formulate our methodology for learning similarity metrics that target dissipative physical systems, we turn to the fundamental quantity of entropy. The second law of thermodynamics states that the entropy S of a closed physical system never decreases, thus $\Delta S \geq 0$. In the following, we make the reasonable assumption that the behavior of the system is continuous and non-oscillating, and that $\Delta S > 0$.² Eq. 1 is the Boltzmann equation from statistical mechanics (Boltzmann 1866), that describes S in terms of the Boltzmann constant k_B and the number of microstates W of a system.³

$$S = k_B \log(W) \quad (1)$$

Since entropy only depends on a single system state, it can be reformulated to take the relative change between two states into account. From an information-theoretical perspective, this is related to using Shannon entropy (Shannon 1948) as a diversity measure, as done by Rényi (1961). Given a sequence of states s_0, s_1, \dots, s_n , we define the relative entropy

$$\tilde{S}(\mathbf{s}) = k \log(10^c \mathbf{w}_s). \quad (2)$$

Here, \mathbf{w}_s is the monotonically increasing, relative number of microstates defined as 0 for s_0 and as 1 for s_n . $10^c > 0$ is a system-dependent factor that determines how quickly the number of microstates increases, i.e. it represents the speed at which different processes decorrelate. As the properties of similarity metrics dictate that distances are always non-negative and only zero for identical states, the lower bound in Eq. 2 is adjusted to 0, leading to a first similarity model $\hat{D}(\mathbf{s}) = k \log(10^c \mathbf{w}_s + 1)$. Finally, relative similarities are equivalent up to a multiplicative constant, and thus we can freely choose k . Choosing $k = 1/(\log 10^c + 1)$ leads to the full similarity model

$$D(\mathbf{s}) = \frac{\log(10^c \mathbf{w}_s + 1)}{\log(10^c + 1)}. \quad (3)$$

For a sequence \mathbf{s} , it predicts the overall similarity behavior between the reference s_0 and the other states with respect to entropy, given the relative number of microstates \mathbf{w}_s and the system decorrelation speed c .

Fig. 1 illustrates the connection between the logarithmically increasing entropy and the proposed similarity model

²These assumptions are required to create sequences with meaningful ground truth distances below in Sec. 4.

³We do not have any a priori information about the distribution of the likelihood of each microstate in a general physical system. Thus, the Boltzmann entropy definition which assumes a uniform microstate distribution is used in the following, instead of more generic entropy models such as the Gibbs or Shannon entropy.

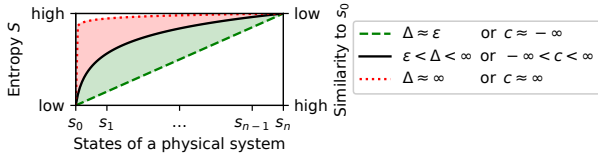


Figure 1: Idealized model of the behavior of entropy and similarity for different Δ or different c , respectively.

for a state sequence with fixed length n . Here, Δ denotes the magnitude of change between the individual sequence states which is directly related to w_s , and c is the decorrelation speed of the system that produced the sequence. In the following, we will refer to the property of a sequence being informative with respect to a pairwise similarity analysis as *difficulty*. Sequences that align with the red dotted curve contain little information as they are dissimilar to s_0 too quickly, either because the original system decorrelates too fast or because the difference between each state is too large (*high difficulty*). On the other hand, sequences like the green dashed curve are also not ideal as they change too little, and a larger non-practical sequence length would be necessary to cover long-term effects (*low difficulty*). Ideally, a sequence s employed for learning tasks should evenly exhibit both regimes as well as intermediate ones, as indicated by the black curve. The central challenges now become finding sequences with a suitable magnitude of Δ , determining c , and assigning distances d to pairs from the sequence.

4 Sequence Creation

To create a sequence s_0, s_1, \dots, s_n within a controlled environment, we make use of the knowledge about the underlying physical processes: We either employ direct changes, based on spatial or temporal coherences to s_0 , or use changes to the initial conditions of the process that lead to s_0 . As we can neither directly determine c nor d at this point, we propose to use proxies for them during the sequence generation. Initially, this allows for finding sequences that roughly fall in a suitable difficulty range, and accurate values can be computed afterwards. Here, we use the mean squared error (MSE) as a proxy distance function and the PCC to determine c , to iteratively update Δ to a suitable range.

Given any value of Δ and a corresponding sequence, pairwise proxy distances⁴ between the sequence elements are computed $d^\Delta = \text{MSE}(s_i, s_j)$ and min-max normalized to $[0, 1]$. Next, we determine a distance sequence corresponding to the physical changes over the states, which we model as a simple linear increase over the sequence $w_s = (j - i)/n$ following (Kohl, Um, and Thuerey 2020). To indirectly determine c , we compare how both distance sequences differ in terms of the PCC as $r = \text{PCC}(d^\Delta, w_s)$. We empirically determined that correlations between 0.65 and 0.85 work well for all cases we considered. In practice, the network stops learning effectively for lower correlation values as states are

⁴To keep the notation clear and concise, sequentially indexing the distance vectors d^Δ and w_s with i and j is omitted here.

too different, while sequences with higher values reduce generalization as a simple metric is sufficient to describe them. Using these thresholds, we propose two semi-automatic iterative methods to create data, depending on the method to introduce variations to a given state (see Fig. 2). Both methods sample a small set of sequences to calibrate Δ to a suitable magnitude and use that value for the full data set. Compared to strictly sampling every sequence, this method is computationally significantly more efficient as less sampling is needed, and it results in a more natural data distribution.

[A] Variations from Initial Conditions of Simulations Given a numerical PDE solver and a set of initial conditions or parameters p , the solver computes a solution to the PDE over the time steps t_0, t_1, \dots, t_t . To create a larger number of different sequences, we make the systems non-deterministic by adding noise to a simulation field and randomly generating the initial conditions from a given range. Adjusting *one* of the parameters p_i in steps with a small perturbation Δ_i , allows for the creation of a sequence s_0, s_1, \dots, s_n with decreasing similarity to the unperturbed simulation output s_0 . This is repeated for every suitable parameter in p , and the corresponding Δ is updated individually until the targeted MSE correlation range is reached. The global noise strength factor also influences the difficulty and can be updated.

[B] Variations from Spatio-temporal Coherences For a source D of volumetric spatio-temporal data without access to a solver, we rely on a larger spatial and/or temporal dimension than the one required for a sequence. We start at a random spatio-temporal position p to extract a cubical spatial area s_0 around it. p can be repeatedly translated in space and/or time by $\Delta_{t,x,y,z}$ to create a sequence s_0, s_1, \dots, s_n of decreasing

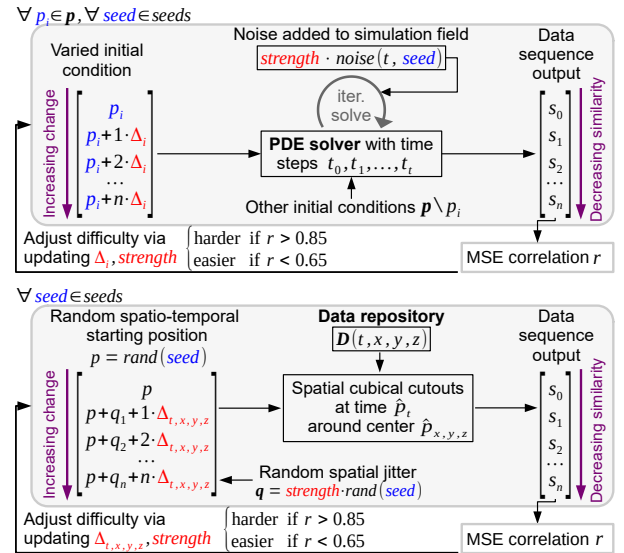


Figure 2: Iteration schemes to calibrate and create data sequences of decreasing similarity. Variation from the reference state can be introduced via the initial conditions of a numerical PDE simulation (method [A], top), or via spatio-temporal data changes on data from a repository (method [B], bottom).

similarity. Note that individual translations in space or time should be preferred if possible. Using different starts leads to new sequences, as long as enough diverse data is available. It is possible to add some global random perturbations q to the positions to further increase the difficulty.

Data Sets To create training data with method [A], we utilize solvers for a basic Advection-Diffusion model (Adv), Burgers’ equation (Bur) with an additional viscosity term, and the full Navier-Stokes equations via a Eulerian smoke simulation (Smo) and a hybrid Eulerian-Lagrangian liquid simulation (Liq). The corresponding validation sets are generated with a separate set of random seeds. Furthermore, we use adjusted versions of the noise integration for two test sets, by adding noise to the density instead of the velocity in the Advection-Diffusion model (AdvD) and overlaying background noise in the liquid simulation (LiqN).

We create seven test sets via method [B]. Four come from the Johns Hopkins Turbulence Database JHTDB (Perlman et al. 2007) that contains a large amount of direct numerical simulation (DNS) data, where each is based on a subset of the JHTDB and features different characteristics: isotropic turbulence (Iso), a channel flow (Cha), magneto-hydrodynamic turbulence (Mhd), and a transitional boundary layer (Tra). Since turbulence contains structures of interest across all length scales, we additionally randomly stride or interpolate the query points for scale factors in $[0.25, 4]$ to create sequences of different physical size. One additional test set (SF) via temporal translations is based on ScalarFlow (Eckert, Um, and Thuerey 2019), consisting of 3D reconstructions of real smoke plumes. Furthermore, method [B] is slightly modified for two synthetic test sets: Instead of using a data repository, we procedurally synthesize spatial fields: We employ linearly moving randomized shapes (Sha), and randomized damped waves (Wav) of the general form $f(x) = \cos(x) * e^{-x}$. All data was gathered in sequences with $n = 10$ at resolution 128^3 , and downsampled to 64^3 for computational efficiency during training and evaluations.

Determining c For each calibrated sequence, we can now more accurately estimate c . As c corresponds to the decorrelation speed of the system, we choose Pearson’s distance $d_i^\Delta = 1 - |\text{PCC}(s_0, s_i)|$ as a distance proxy here. c is determined via standard unbounded least-squares optimization from the similarity model in Eq. 3 as $c = \arg \min_c \frac{\log(10^c d^\Delta + 1)}{\log(10^c + 1)}$.

5 Learning a Distance Function

Given the calibrated sequences s of different physical systems with elements s_0, s_1, \dots, s_n , the corresponding value of c , and the pairwise physical target distance sequence $w_s = (j - i)/n$, we can now formulate a semi-supervised learning problem: We train a neural network m that receives pairs from s as an input, and outputs scalar distances d for each pair. These predictions are trained against ground truth distances $g = \frac{\log(10^c w_s + 1)}{\log(10^c + 1)}$. Note that g originates from the sequence order determined by our data generation approach, transformed with a non-linear transformation according to the entropy-based similarity model. This technique incorporates the underlying physical behavior by accounting for the

decorrelation speed over the sequence, compared to adding variations in a post-process (as commonly done in the domain of images, e.g. by Ponomarenko et al. (2015)). To train the metric network, the correlation loss function in Eq. 4 below compares d to g and provides gradients.

Network Structure For our method, we generally follow the established Siamese network structure, that was originally proposed for 2D domains (Zhang et al. 2018): First, two inputs are embedded in a latent space using a CNN as a feature extractor. The Siamese structure means that the weights are shared, which ensures the mathematical requirements for a pseudo-metric (Kohl, Um, and Thuerey 2020). Next, the features from all layers are normalized and compared with an element-wise comparison like an absolute or squared difference. Finally, this difference is aggregated with sum, mean, and learned weighted average functions. To compute the proposed *VolSiM* metric that compares inherently more complex 3D data, changes to this framework are proposed below.

Multiscale Network Scale is important for a reliable similarity assessment, since physical systems often exhibit self-similar behavior that does not significantly change across scales, as indicated by the large number of dimensionless quantities in physics. Generally, scaling a data pair should not alter its similarity, and networks can learn such an invariance to scale most effectively by processing data at different scales. One example where this is crucial is the energy cascade in turbulence (Pope 2000), which is also analyzed in our case study below. For learned image metrics, this invariance is also useful (but less crucial), and often introduced with large strides and kernels in the convolutions, e.g. via a feature extractor based on AlexNet (Zhang et al. 2018). In fact, our experiments with similar architectures showed, that models with large strides and kernels generally perform better than models that modify the scale over the course of the network to a lesser extent. However, we propose to directly encode this scale structure in a multiscale architecture for a more accurate similarity assessment, and a network with a smaller resource footprint.

Fig. 3 shows the proposed fully convolutional network: Four scale blocks individually process the input on increasingly smaller scales, where each block follows the same layer structure, but deeper blocks effectively cover a significantly larger volume due to the reduced input resolutions. Deeper

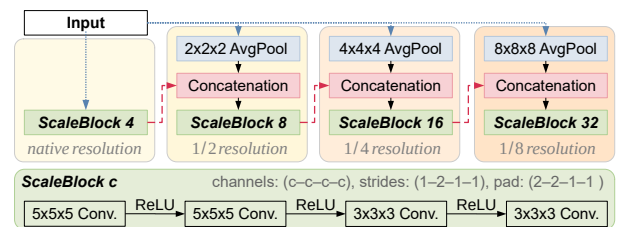


Figure 3: Standard Conv+ReLU blocks (bottom) are interwoven with input and resolution connections (blue dotted and red dashed), to form the combined network architecture (top) with about 350k weights.

architectures can model complex functions more easily, so we additionally include resolution connections from each scale block to the next resolution level via concatenation. Effectively, the network learns a mixture of connected deep features and similar representations across scales as a result.

Training and Evaluation To increase the model’s robustness during training, we used the following data augmentations for each sequence: the data is normalized to $[-1, 1]$, and together randomly flipped and rotated in increments of 90° around a random axis. The velocity channels are randomly swapped to prevent directional biases from some simulations, while scalar data is extended to the three input channels via repetition. For inference, only the normalization operation and the repetition of scalar data is performed. The final metric model was trained with the Adam optimizer with a learning rate of 10^{-4} for 30 epochs via early stopping. To determine the accuracy of any metric during inference in the following, we compute the SRCC between the distance predictions of the metric \mathbf{d} and the ground truth \mathbf{w}_s , where a value closer to 1 indicates a better reconstruction.⁵

Loss Function Given predicted distances \mathbf{d} and a ground truth \mathbf{g} of size n , we train our metric networks with the loss

$$L(\mathbf{d}, \mathbf{g}) = \lambda_1(\mathbf{d} - \mathbf{g})^2 + \lambda_2(1 - r)$$

$$\text{where } r = \frac{\sum_{i=1}^n (d_i - \bar{d})(g_i - \bar{g})}{\sqrt{\sum_{i=1}^n (d_i - \bar{d})^2} \sqrt{\sum_{i=1}^n (g_i - \bar{g})^2}}. \quad (4)$$

consisting of a weighted combination of an MSE and an inverted correlation term r , where \bar{d} and \bar{g} denote the mean. While the formulation follows existing work (Kohl, Um, and Thuerey 2020), it is important to note that \mathbf{g} is computed by our similarity model from Sec. 3, and below we introduce a slicing technique to apply this loss formulation to high-dimensional data sets.

To successfully train a neural network, Eq. 4 requires a trade-off: A large batch size b is useful to improve training stability via less random gradients for optimization. Similarly, a sufficiently large value of n is required to keep the correlation values accurate and stable. However, with finite amounts of memory, choosing large values for n and b is not possible in practice. Especially so for 3D cases, where a single sample can already be memory intensive. In general, n is implicitly determined by the length of the created sequences via the number of possible pairs. Thus, we provide an analysis how the correlation can be approximated in multiple steps for a fixed n , to allow for increasing b in return. In the following, the batch dimension is not explicitly shown, but all expressions can be extended with a vectorized first dimension. The full distance vectors \mathbf{d} and \mathbf{g} are split in slices with v elements, where v should be a proper divisor of n . For any slice k , we can compute a partial correlation r_k with

$$r_k = \frac{\sum_{i=k}^{k+v} (d_i - \bar{d})(g_i - \bar{g})}{\sqrt{\sum_{i=k}^{k+v} (d_i - \bar{d})^2} \sqrt{\sum_{i=k}^{k+v} (g_i - \bar{g})^2}}. \quad (5)$$

⁵This is equivalent to SRCC(\mathbf{d}, \mathbf{g}), since the SRCC measures monotonic relationships and is not affected by monotonic transformations, but using \mathbf{w}_s is more efficient and has numerical benefits.

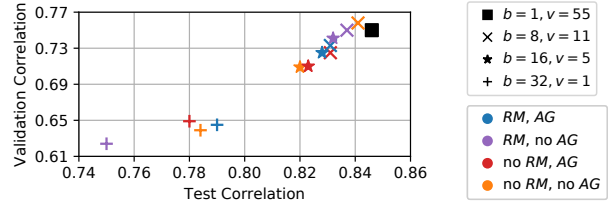


Figure 4: Combined validation and test performance for different batch sizes b and slicing values v (markers), and the usage of running sample mean RM and correlation aggregation AG (colors).

Note that this is only an approximation, and choosing larger values of v for a given b is always beneficial, if sufficient memory is available. For all slices, the gradients are accumulated during backpropagation since other aggregations would require a computational graph of the original, impractical size. Eq. 5 still requires the computation of the means \bar{d} and \bar{g} as a pre-process over all samples. Both can be approximated with the running means \tilde{d} and \tilde{g} for efficiency (RM). For small values of v , the slicing results in very coarse, unstable correlation results. To alleviate that, it is possible to use a running mean over all previous values $\tilde{r}_k = (1/k)(r_k + \sum_{l=1}^{k-1} r_l)$. This aggregation (AG) can stabilize the gradients of individual r_k as they converge to the true correlation value.

Fig. 4 displays the resulting performance on our data, when training with different combinations of b , v , RM , and AG . All models exhibit similar levels of memory consumption and were trained with the same random training seed. When comparing models with and without RM both are on par in most cases, even though computation times for a running mean are about 20% lower. Networks with and without AG generalize similarly, however, models with the aggregation exhibit less fluctuations during optimization, leading to an easier training process. Overall, this experiment demonstrates that choosing larger v consistently leads to better results (marker shape), so more accurate correlations are beneficial over a large batch size b in memory-limited scenarios. Thus, we use $b = 1$ and $v = 55$ for the final model.

6 Results

We compare the proposed $VolSiM$ metric to a variety of existing methods in the upper section of Tab. 1. All metrics were evaluated on the volumetric data from Sec. 4, which contain a wide range of test sets that differ strongly from the training data. $VolSiM$ consistently reconstructs the ground truth distances from the entropy-based similarity model more reliably than other approaches on most data sets. As expected, this effect is most apparent on the validation sets since their distribution is closest to the training data. But even on the majority of test sets with a very different distribution, $VolSiM$ is the best performing or close to the best performing metric. Metrics without deep learning often fall short, indicating that they were initially designed for different use cases, like $SSIM$ (Wang et al. 2004) for images, or variation

Metric	Validation data sets				Test data sets										
	Simulated				Simulated		Generated		JHTDB ^a				SF ^b	^c	
	Adv	Bur	Liq	Smo	AdvD	LiqN	Sha	Wav	Iso	Cha	Mhd	Tra	SF	All	
<i>MSE</i>	0.61	0.70	0.51	0.68	0.77	0.76	0.75	0.65	0.76	0.86	0.80	0.79	0.79	0.70	
<i>PSNR</i>	0.61	0.68	0.52	0.68	0.78	0.76	0.75	0.65	0.78	0.86	0.81	0.83	0.79	0.73	
<i>SSIM</i>	0.75	0.68	0.49	0.64	0.81	0.80	0.76	0.88	0.49	0.55	0.62	0.60	0.44	0.61	
<i>VI</i>	0.57	0.69	0.43	0.60	0.69	0.82	0.67	0.87	0.59	0.76	0.68	0.67	0.41	0.62	
<i>LPIPS (2D)</i>	0.63	0.62	0.35	0.56	0.76	0.62	0.87	0.92	0.71	0.83	0.79	0.76	0.87	0.76	
<i>LSiM (2D)</i>	0.57	0.55	0.48	0.71	0.79	0.75	0.93	0.97	0.69	0.86	0.79	0.81	0.98	0.81	
<i>VolSiM (ours)</i>	0.75	0.73	0.66	0.77	0.84	0.88	0.95	0.96	0.77	0.86	0.81	0.88	0.95	0.85	
<i>CNN_{trained}</i>	0.60	0.71	0.63	0.76	0.81	0.77	0.92	0.93	0.75	0.86	0.78	0.85	0.95	0.82	
<i>MS_{rand}</i>	0.57	0.66	0.45	0.69	0.76	0.75	0.80	0.78	0.74	0.86	0.80	0.82	0.84	0.74	
<i>CNN_{rand}</i>	0.52	0.66	0.49	0.69	0.77	0.70	0.93	0.96	0.74	0.85	0.79	0.83	0.95	0.81	
<i>MS_{identity}</i>	0.75	0.71	0.68	0.73	0.83	0.85	0.87	0.96	0.74	0.87	0.77	0.87	0.94	0.82	
<i>MS_{3 scales}</i>	0.70	0.69	0.70	0.73	0.83	0.82	0.95	0.94	0.76	0.87	0.80	0.88	0.93	0.83	
<i>MS_{5 scales}</i>	0.78	0.72	0.78	0.78	0.81	0.90	0.94	0.93	0.75	0.85	0.77	0.88	0.93	0.82	
<i>MS_{added Iso}</i>	0.73	0.72	0.77	0.79	0.84	0.84	0.92	0.97	[0.79]	0.87	0.80	0.86	0.97	0.84	
<i>MS_{only Iso}</i>	0.58	0.62	0.32	0.63	0.78	0.65	0.72	0.92	[0.82]	0.77	0.86	0.79	0.65	0.75	

^a Johns Hopkins Turbulence DB (Perlman et al. 2007) ^b ScalarFlow (Eckert, Um, and Thuerey 2019) ^c Combined test data sets

Table 1: Top: performance comparison of different metrics for 3D data via the SRCC, where values closer to 1 indicate a better reconstruction of the ground truth distances (bold+underlined: best method for each data set, underlined: within a 0.01 margin of the best performing). Bottom: ablation study of the proposed method (brackets: advantage due to different training data).

of information *VI* (Meilä 2007) for clustering. The strictly element-wise metrics *MSE* and *PSNR* exhibit almost identical performance, and both work poorly on a variety of data sets. As the learning-based methods *LPIPS* (Zhang et al. 2018) and *LSiM* (Kohl, Um, and Thuerey 2020) are limited to two dimensions, their assessments in Tab. 1 are obtained by averaging sliced evaluations for all three spatial axes. Both methods show improvements over the element-wise metrics, but are still clearly inferior to the performance of *VolSiM*. This becomes apparent on our aggregated test sets displayed in the **All** column, where *LSiM* results in a correlation value of 0.81, compared to *VolSiM* with 0.85. *LSiM* can only come close to *VolSiM* on less challenging data sets where correlation values are close to 1 and all learned reconstructions are already highly accurate. This improvement is comparable to using *LPIPS* over *PSNR*, and represents a significant step forward in terms of a robust similarity assessment.

The bottom half of Tab. 1 contains an ablation study of the proposed architecture *MS*, and a simple *CNN* model. This model is similar to an extension of the convolution layers of AlexNet (Krizhevsky, Sutskever, and Hinton 2017) to 3D, and does not utilize a multiscale structure. Even though *VolSiM* has more than 80% fewer weights compared to *CNN_{trained}*, it can fit the training data more easily and generalizes better for most data sets in Tab. 1, indicating the strengths of the proposed multiscale architecture. The performance of untrained models *CNN_{rand}* and *MS_{rand}* confirm the findings from Zhang et al. (2018), who also report a surprisingly strong performance of random networks. We replace the non-linear transformation of w_s from the similarity model with an identity transformation for *MS_{identity}* during training, i.e. only the sequence order determines g . This consistently lowers the generalization of the metric across data sets, indicating that

well calibrated sequences as well as the similarity model are important for the similarity assessment. Removing the last resolution scale block for *MS_{3 scales}* overly reduces the capacity of the model, while adding another block for *MS_{5 scales}* is not beneficial. In addition, we also investigate two slightly different training setups: for *MS_{added Iso}* we integrate additional sequences created like the **Iso** data in the training, while *MS_{only Iso}* is exclusively trained on such sequences. *MS_{added Iso}* only slightly improves upon the baseline, and even the turbulence-specific *MS_{only Iso}* model does not consistently improve the results on the JHTDB data sets. Both cases indicate a high level of generalization for *VolSiM*, as it was not trained on any turbulence data.

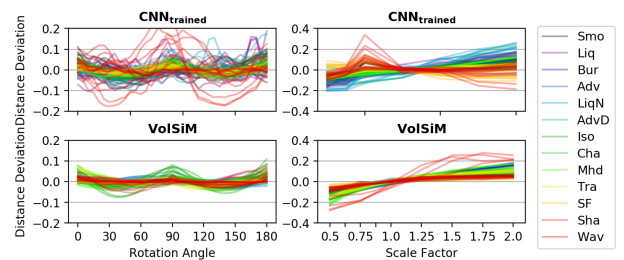


Figure 5: Distance deviation from the mean prediction over differently rotated (left) and scaled (right) inputs for a simple CNN and the proposed multiscale model.

Transformation Invariance Physical systems are often characterized by Galilean invariance (McComb 1999), i.e. identical laws of motion across inertial frames. Likewise, a metric should be invariant to transformations of the input, meaning a constant distance output when translating,

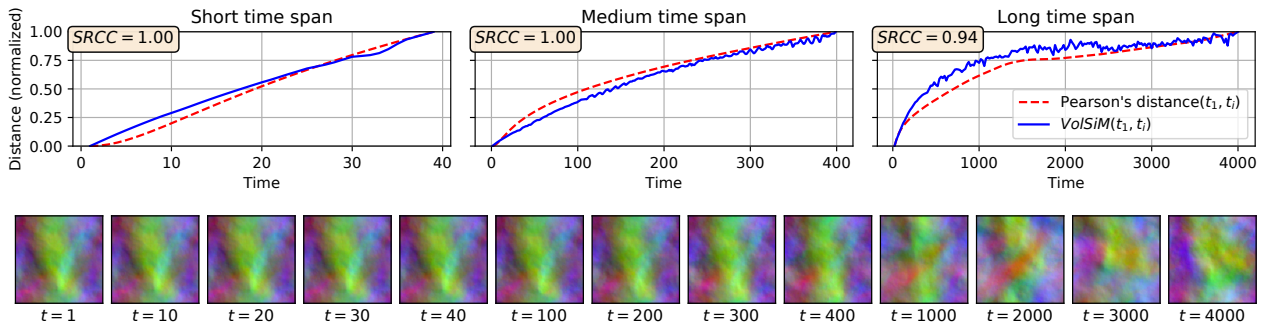


Figure 6: Top: Analysis of forced isotropic turbulence across three time spans. The high SRCC values indicate strong agreement between a traditional correlation evaluation and *VolSiM*. Bottom: Examples from the sequence, visualized via a mean projection along the x-axis and color-coded channels.

scaling, or rotating both inputs. Element-wise metrics fulfill these properties by construction, but our Siamese network structure requires an equivariant feature representation that changes along with input transformations to achieve them. As CNN features are translation equivariant by design (apart from boundary effects and pooling), we empirically examine rotation and scale invariance for our multiscale metric and a standard Conv+ReLU model on a fixed set of 8 random data pairs from each data set. For the rotation experiment, we rotate the pairs in steps of 5° around a random coordinate axis. The empty volume inside the original frame is filled with a value of 0, and data outside the frame is cropped. For scaling, the data is bilinearly up- or downsampled according to the scale factor, and processed fully convolutionally.

In Fig. 5, the resulting distance deviation from the mean of the predicted distances is plotted for rotation and scaling operations. The optimal result would be a perfectly equal distance with zero deviation across all transformations. Compared to the model $CNN_{trained}$, it can be observed that *VolSiM* produces less deviations overall, and leads to significantly smoother and more consistent distance curves, across scales and rotations as shown in Fig. 5. This is caused by the multiscale architecture, which results in a more robust internal feature representation, and thus higher stability across small transformations. Note that we observe scale equivariance rather than invariance for *VolSiM*, i.e. a mostly linear scaling of the distances according to the input size. This is most likely caused by a larger spatial size of the fully convolutional features. Making a scale equivariant model fully invariant would require a form of normalization, which is left for future work.

Case Study: Turbulence Analysis As a particularly challenging test for generalization, we further perform a case study on forced isotropic turbulence that resembles a potential real-world scenario for our metric in Fig. 6. For this purpose, fully resolved raw DNS data over a long temporal interval from the isotropic turbulence data set from JHTDB is utilized (see bottom of Fig. 6). The 1024^3 domain is filtered and reduced to a size of 128^3 via strides, meaning *VolSiM* is applied in a fully convolutional manner, and has to generalize beyond the training resolution of 64^3 . Three different time spans of the simulation are investigated, where the long span

also uses temporal strides. Traditionally, turbulence research makes use of established two-point correlations to study such cases (Pope 2000). Since we are interested in a comprehensive spatial analysis instead of two single points, we can make use of Pearson’s distance that corresponds to an aggregated two-point correlation on the full fields to obtain a physical reference evaluation in this scenario.

Fig. 6 displays normalized distance values between the first simulation time step t_1 and each following step t_i . Even though there are smaller fluctuations, the proposed *VolSiM* metric (blue) behaves very similar to the physical reference of aggregated two-point correlations (red dashed) across all time spans. This is further emphasized by the high SRCC values between both sets of trajectories, even for the challenging long time span. Our metric faithfully recovers the correlation-based reference, despite not having seen any turbulence data at training time. Overall, this experiment shows that the similarity model integrates physical concepts into the comparisons of *VolSiM*, and indicates the generalization capabilities of the multiscale metric to new cases.

7 Conclusion

We presented the multiscale CNN architecture *VolSiM*, and demonstrated its capabilities as a similarity metric for volumetric simulation data. A similarity model based on the behavior of entropy in physical systems was proposed and utilized to learn a robust, physical similarity assessment. Different methods to compute correlations inside a loss function were analyzed, and the invariance to scale and rotation transformations investigated. Furthermore, we showed clear improvements upon elementwise metrics as well as existing learned approaches like *LPIPS* and *LSiM* in terms of an accurate similarity assessment across our data sets.

The proposed metric potentially has an impact on a broad range of disciplines where volumetric simulation data arises. An interesting area for future work is designing a metric specifically for turbulence simulations, first steps towards which were taken with our case study. Additionally, investigating learning-based methods with features that are by construction equivariant to rotation and scaling may lead to further improvements in the future.

Ethical Statement

Since we target the fundamental problem of the similarity assessment of numerical simulations, we do not see any direct negative ethical implications of our work. However, there could be indirect negative effects since this work can act as a tool for more accurate and/or robust numerical simulations in the future, for which a military relevance exists. A further indirect issue could be explainability, e.g. when simulations in an engineering process yield unexpected inaccuracies.

Acknowledgments

This work was supported by the ERC Consolidator Grant *SpaTe* (CoG-2019-863850).

References

- Andersson, P.; Nilsson, J.; Akenine-Möller, T.; Oskarsson, M.; Åström, K.; and Fairchild, M. D. 2020. FLIP: A Difference Evaluator for Alternating Images. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2): 15:1–15:23.
- Avgoustinakis, P.; Kordopatis-Zilos, G.; Papadopoulos, S.; Symeonidis, A. L.; and Kompatsiaris, I. 2020. Audio-Based Near-Duplicate Video Retrieval with Audio Similarity Learning. In *25th International Conference on Pattern Recognition (ICPR 2020)*, 5828–5835.
- Bell, S.; and Bala, K. 2015. Learning Visual Similarity for Product Design with Convolutional Neural Networks. *ACM Transactions on Graphics*, 34(4): 98:1–98:10.
- Boltzmann, L. 1866. *Über Die Mechanische Bedeutung Des Zweiten Hauptsatzes Der Wärmetheorie: (Vorgelegt in Der Sitzung Am 8. Februar 1866)*. Staatsdruckerei.
- Bosse, S.; Maniry, D.; Mueller, K.-R.; Wiegand, T.; and Samek, W. 2016. Neural Network-Based Full-Reference Image Quality Assessment. In *2016 Picture Coding Symposium (PCS)*.
- Chidester, B.; Zhou, T.; Do, M. N.; and Ma, J. 2019. Rotation Equivariant and Invariant Neural Networks for Microscopy Image Analysis. *Bioinformatics*, 35(14): i530–i537.
- Chopra, S.; Hadsell, R.; and LeCun, Y. 2005. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 539–546. IEEE Computer Society.
- Chu, M.; and Thurey, N. 2017. Data-Driven Synthesis of Smoke Flows with CNN-Based Feature Descriptors. *ACM Transactions on Graphics*, 36(4): 69:1–69:14.
- Dosovitskiy, A.; and Brox, T. 2016. Generating Images with Perceptual Similarity Metrics Based on Deep Networks. In *Advances in Neural Information Processing Systems 29*, volume 29.
- Eckert, M.-L.; Um, K.; and Thurey, N. 2019. ScalarFlow: A Large-Scale Volumetric Data Set of Real-World Scalar Transport Flows for Computer Animation and Machine Learning. *ACM Transactions on Graphics*, 38(6).
- Hadsell, R.; Chopra, S.; and LeCun, Y. 2006. Dimensionality Reduction by Learning an Invariant Mapping. In *2006 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 1735–1742.
- Holmes, P.; Lumley, J. L.; Berkooz, G.; and Rowley, C. W. 2012. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press. ISBN 978-0-511-91970-1.
- Horé, A.; and Ziou, D. 2010. Image Quality Metrics: PSNR vs. SSIM. In *20th International Conference on Pattern Recognition (ICPR 2010)*, 2366–2369.
- Huynh-Thu, Q.; and Ghanbari, M. 2008. Scope of Validity of PSNR in Image/Video Quality Assessment. *Electronics Letters*, 44(13): 800–801.
- Huynh-Thu, Q.; and Ghanbari, M. 2012. The Accuracy of PSNR in Predicting Video Quality for Different Video Scenes and Frame Rates. *Telecommunication Systems*, 49(1): 35–48.
- Johnson, J.; Alahi, A.; and Fei-Fei, L. 2016. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Computer Vision - ECCV 2016*, volume 9906, 694–711.
- Kohl, G.; Um, K.; and Thurey, N. 2020. Learning Similarity Metrics for Numerical Simulations. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119, 5349–5360.
- Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60(6): 84–90.
- Kumari, P.; Chaudhuri, S.; and Chaudhuri, S. 2019. PerceptNet: Learning Perceptual Similarity of Haptic Textures in Presence of Unorderable Triplets. In *2019 IEEE World Haptics Conference (WHC)*, 163–168. IEEE.
- McComb, W. D. 1999. *Dynamics and Relativity*. Oxford University Press. ISBN 0-19-850112-9.
- Meilä, M. 2007. Comparing Clusterings—an Information Based Distance. *Journal of Multivariate Analysis*, 98(5): 873–895.
- Nilsson, J.; and Akenine-Möller, T. 2020. Understanding SSIM. *arXiv:2006.13846 [cs, eess]*.
- Olufsen, M. S.; Peskin, C. S.; Kim, W. Y.; Pedersen, E. M.; Nadim, A.; and Larsen, J. 2000. Numerical Simulation and Experimental Validation of Blood Flow in Arteries with Structured-Tree Outflow Conditions. *Annals of Biomedical Engineering*, 28(11): 1281–1299.
- Pearson, K. 1920. Notes on the History of Correlation. *Biometrika*, 13(1): 25–45.
- Perlman, E.; Burns, R.; Li, Y.; and Meneveau, C. 2007. Data Exploration of Turbulence Simulations Using a Database Cluster. In *Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing*, 1–11.
- Ponomarenko, N.; Jin, L.; Ieremeiev, O.; Lukin, V.; Egiazarian, K.; Astola, J.; Vozel, B.; Chehdi, K.; Carli, M.; Battisti, F.; and Kuo, C. C. J. 2015. Image Database TID2013: Peculiarities, Results and Perspectives. *Signal Processing-Image Communication*, 30: 57–77.
- Pope, S. 2000. *Turbulent Flows*. Cambridge University Press. ISBN 978-0-511-84053-1.

Prashnani, E.; Cai, H.; Mostofi, Y.; and Sen, P. 2018. PieAPP: Perceptual Image-Error Assessment through Pairwise Preference. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1808–1817. IEEE Computer Society.

Rényi, A. 1961. On Measures of Entropy and Information. In *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, 547–561. University of California Press.

Roth, K.; Milbich, T.; Sinha, S.; Gupta, P.; Ommer, B.; and Cohen, J. P. 2020. Revisiting Training Strategies and Generalization Performance in Deep Metric Learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119, 8242–8252. PMLR.

Shannon, C. E. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3): 379–423.

Spearman, C. 1904. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology*, 15(1): 72–101.

Stocker, T.; Qin, D.; Plattner, G.-K.; Tignor, M.; Allen, S.; Borschung, J.; Nauels, A.; Xia, Y.; Bex, V.; and Midgley, P. 2014. *Climate Change 2013: The Physical Science Basis*. Cambridge University Press. ISBN 978-1-107-41532-4.

Talebi, H.; and Milanfar, P. 2018. Learned Perceptual Image Enhancement. In *2018 IEEE International Conference on Computational Photography (ICCP)*.

Thuerey, N.; Holl, P.; Mueller, M.; Schnell, P.; Trost, F.; and Um, K. 2021. Physics-Based Deep Learning. <https://physicsbaseddeeplearning.org>. Accessed: 2023-03-27.

Um, K.; Hu, X.; and Thuerey, N. 2017. Perceptual Evaluation of Liquid Simulation Methods. *ACM Transactions on Graphics*, 36(4).

Um, K.; Hu, X.; Wang, B.; and Thuerey, N. 2021. Spot the Difference: Accuracy of Numerical Simulations via the Human Visual System. *ACM Transactions on Applied Perception*, 18(2): 6:1–6:15.

Wang, R.; Walters, R.; and Yu, R. 2021. Incorporating Symmetry into Deep Dynamics Models for Improved Generalization. In *9th International Conference on Learning Representations (ICLR 2021)*.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612.

Weiler, M.; Geiger, M.; Welling, M.; Boomsma, W.; and Cohen, T. 2018. 3D Steerable CNNs: Learning Rotationally Equivariant Features in Volumetric Data. In *Advances in Neural Information Processing Systems 31*, 10402–10413.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 586–595.

APPENDIX: Learning Similarity Metrics for Volumetric Simulations with Multiscale CNNs

In the following, additional details for the proposed *VolSiM* metric are provided: App. A contains implementation details regarding the training and the metric model setup, and App. B features generation details and visualizations for all our data sets, as well as an analysis of the data distribution resulting from the data generation. The training stability for the loss experiment is investigated in App. C, and details for the experimental setup of the ablation study models are given in App. D. Afterwards, App. E contains additional ablation studies, and details regarding the turbulence case study can be found in App. F.

A Implementation Details

The training and evaluation process of the metric was implemented in PyTorch (Paszke et al. 2019), while the data was simulated and collected with specialized solvers and data interfaces as described in App B. The data acquisition, training, and metric evaluation was performed on a server with an Intel i7-6850 (3.60Ghz) CPU and an NVIDIA GeForce GTX 1080 Ti GPU. It took about 38 hours of training to fully optimize the final *VolSiM* model for the data sequences with a spatial resolution of 64^3 .

In addition to the multiscale feature extractor network, the following operations were used for the Siamese architecture of the metric: Each feature map is normalized via a mean and standard deviation normalization to a standard normal distribution. The mean and standard deviation of each feature map is computed in a pre-processing step for the initialization of the network over all data samples. Both values are fixed for training the metric afterwards. To compare both sets of feature maps in the latent space, a simple element-wise, squared difference is employed. To keep the mathematical metric properties, this also requires a square root operation before the final distance output. The spatial squared feature map differences are then aggregated along all dimensions into a scalar distance output. Here, we used a single learned weight with dropout for every feature map, to combine them to a weighted average per network layer. The activations of the average feature maps are spatially combined with a simple mean, and summed over all network layers afterwards. This process of normalizing, comparing, and aggregating the feature maps computed by the feature extractor follows previous work (Kohl, Um, and Thuerey 2020; Zhang et al. 2018).

The weights to adjust the influence of each feature map are initialized to 0.1, all other weights of the multiscale feature extractor are initialized with the default PyTorch initialization. For the final loss, the MSE term was weighted with $\lambda_1 = 1.0$, while the correlation term was weighted with $\lambda_2 = 0.7$.

B Data Set Details

In the following sections, the details underlying each data set are described. Tab. 3 contains a summary of simulator, simulation setup, varied parameters, noise integration, and used fields for the simulated and generated data sets. Tab. 4

features a summary of the collected data sets, with repository details, jitter and cutout settings, and spatial and temporal Δ values. Both tables also contain the number of sequences created for training, validation, and testing for every data source. These values only apply for the general metric setup, and changes for the ablation study models can be found in App. D below.

B.1 Advection-Diffusion and Burgers' Equation

In its simplest form, the transport of matter in a flow can be described by the two phenomena of advection and diffusion. Advection describes the movement of a passive quantity inside a velocity field over time, and diffusion describes the process of dissipation of this quantity due to the second law of thermodynamics.

$$\frac{\partial d}{\partial t} = \nu \nabla^2 d - u \cdot \nabla d \quad (6)$$

Eq. 6 is the simplified Advection-Diffusion equation with constant diffusivity and no sources or sinks, where u denotes the velocity, d is a scalar passive quantity that is transported, and ν is the diffusion coefficient or kinematic viscosity.

Burgers' Equation in Eq. 7 is similar to the Advection-Diffusion equation, but it describes how the velocity field itself changes over time with advection and diffusion. The diffusion term can also be interpreted as a viscosity, that models the resistance of the material to deformations. Furthermore, this variation can develop discontinuities (also called shock waves). Here, u also denotes the velocity and ν the kinematic viscosity or diffusion coefficient.

$$\frac{\partial u}{\partial t} = \nu \nabla^2 u - u \cdot \nabla u \quad (7)$$

To solve both PDEs, the differentiable fluid framework PhiFlow (Holl, Thuerey, and Koltun 2020) was used. The solver utilizes a Semi-Lagrangian advection scheme, and we chose periodic domain boundary conditions to allow for the usage of a Fourier space diffusion solver. We introduced additional continuous forcing to the simulations by adding a force term f to the velocity after every simulation step. Thus, f depends on the time steps t , that is normalized by division of the simulation domain size beforehand. For Adv, Bur, and AdvD, we initialized the fields for velocity, density, and force with multiple layered parameterized sine functions. This leads to a large range of patterns across multiple scales and frequencies when varying the sine parameters.

$$u^x(\mathbf{p}) = \text{sum} \left(\mathbf{f}_1^x + \sum_{i=1}^4 \mathbf{f}_{i+1}^x \right) * \sin(2^i \pi \mathbf{p} + c_i \mathbf{o}_{(i+1 \bmod 2)+1}^x) \quad (8)$$

where $\mathbf{c} = (1, 1, 0.4, 0.3)$

$$f^x(\mathbf{p}, t) = \text{sum} \left(\mathbf{f}_6^x * (1 + \mathbf{f}_6^x * 20) \right. \\ \left. * \sum_{i=1}^4 \mathbf{f}_{i+1}^x * \sin(2^i \pi \tilde{\mathbf{p}} + c_i \mathbf{o}_{(i \bmod 2)+1}^x) \right) \quad (9)$$

where $\tilde{\mathbf{p}} = \mathbf{p} + \mathbf{f}_7^x * 0.5 + \mathbf{f}_7^x * \sin(3t)$
and $\mathbf{c} = (0, 1, 1, 0.7)$

$$d(\mathbf{p}) = \text{sum} \left(\sum_i^{\{x,y,z\}} \sin(\mathbf{f}_d^i * 24\pi p_i + o_d^i) \right) \quad (10)$$

Eq. 8, 9, and 10 show the layered sine functions in $u^x(\mathbf{p})$, $f^x(\mathbf{p})$, and $d(\mathbf{p})$ for a spatial grid position $\mathbf{p} \in \mathbb{R}^3$. The sum operation denotes a sum of all vector elements here, and all binary operations on vectors and scalars use broadcasting of the scalar value to match the dimensions. Eq. 11 shows the definition of the function parameters used above, all of which are randomly sampled based on the simulation seed for more diverse simulations.

$$\begin{aligned} \mathbf{f}_1^x &\sim \mathcal{U}(-0.2, 0.2)^3 & \mathbf{f}_7^x &\sim \mathcal{U}(-0.1, 0.1)^3 \\ \mathbf{f}_2^x &\sim \mathcal{U}(-0.2, 0.2)^3 & \mathbf{f}_d^{x,y,z} &\sim \{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}\}^3 \\ \mathbf{f}_3^x &\sim \mathcal{U}(-0.15, 0.15)^3 & \nu &\sim \mathcal{U}(0.0002, 0.1002) \\ \mathbf{f}_4^x &\sim \mathcal{U}(-0.15, 0.15)^3 & \mathbf{o}_1^x &\sim \mathcal{U}(0, 100)^3 \\ \mathbf{f}_5^x &\sim \mathcal{U}(-0.1, 0.1)^3 & \mathbf{o}_2^x &\sim \mathcal{U}(0, 100)^3 \\ \mathbf{f}_6^x &\sim \mathcal{U}(0.0, 0.1)^3 & \mathbf{o}_d^{x,y,z} &\sim \mathcal{U}(0, 100)^3 \end{aligned} \quad (11)$$

Note that ν was multiplied by 0.1 for `Bur`. The remaining velocity and force components $u^y(\mathbf{p})$, $u^z(\mathbf{p})$, $f^y(\mathbf{p})$, and $f^z(\mathbf{p})$ and corresponding parameters are omitted for brevity here, since they follow the same initialization pattern as $u^x(\mathbf{p})$ and $f^x(\mathbf{p})$. Tab. 3 shows the function parameters that were varied, by using the random initializations and adjusting one of them in linear steps to create a sequence. The main difference between the Advection-Diffusion training data and the test set is the method of noise integration: For `Adv` it is integrated into the simulation velocity, while for `AdvD` it is added to the density field instead. The amount of noise added to the velocity for `Bur` and `Adv` and to the density for `AdvD` was varied in isolation as well.

B.2 Navier-Stokes Equations

The Navier-Stokes Equations fully describe the behavior of fluids like gases and liquids, via modelling advection, viscosity, and pressure effects, as well as mass conservation. Pressure can be interpreted as the force exerted by surrounding fluid mass at a given point, and the conservation of mass means that the fluid resists compression.

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = -\frac{\nabla P}{\rho} + \nu \nabla^2 u + g \quad (12)$$

$$\nabla \cdot u = 0. \quad (13)$$

Eq. 12 describes the conservation of momentum, and Eq. 13 describes mass conservation. Again, u denotes the velocity, P is the pressure, ρ is the fluids density, ν is the kinematic viscosity, and g denotes external forces like gravity.

Smoke To create the smoke data set `SmO`, the fluid framework MantaFlow (Thuerey and Pfaff 2018) that provides a grid-based Eulerian smoke solver for the Navier-Stokes Equations was used. It is based on a Semi-Lagrangian advection scheme, and on the conjugate gradient method as a pressure solver. The simulation setup consists of a cylindrical smoke source at the bottom of the domain with a fixed noise pattern initialization to create more diverse smoke plumes. Furthermore, a constant spherical force field \mathbf{ff} is positioned over the source. This setup allows for a variation of multiple simulation parameters, like the smoke buoyancy, the source position and different force field settings. They include position, rotation, radius and strength. In addition, the amount of added noise to the velocity can also be varied in isolation.

Liquid Both liquid data sets, `Liq` and `LiqN`, were created with a liquid solver in MantaFlow. It utilizes the hybrid Eulerian-Lagrangian fluid implicit particle method (Zhu and Bridson 2005), that combines the advantages of particle and grid-based liquid simulations for reduced numerical dissipation. The simulation setup consists of two liquid cuboids of different shapes, similar to the common breaking dam setup. After 25 simulation time steps a liquid drop is added near the top of the simulation domain, and it falls down on the water surface that is still moving. Here, the external gravity force as well as the drops position and radius are varied to create similarity sequences. As for the smoke data, a modification of the amount of noise added to the velocity was also employed as a varied parameter. The main difference between the liquid training data and the test set is the method of noise integration: For `Liq` it is integrated into the simulation velocity, while for `LiqN` it is overlaid on the simulation background.

B.3 Generated Data

To create the shape data set `Sha` and the wave data set `Wav`, a random number of straight paths are created by randomly generating a start and end point inside the domain. It is ensured that both are not too close to the boundaries and that the path has a sufficient length. The intermediary positions for the sequence are a result of linearly interpolating on these paths. The positions on the path determine the center for the generated objects that are added to an occupancy marker grid. For both data sets, overlapping shapes and waves are combined additively, and variations with and without overlaid noise to the marker grid were created.

Shapes For `Sha`, random shapes (box or sphere) are added to the positions, where the shape's size is a random fraction of the path length, with a minimum and maximum constraint. The created shapes are then applied to the marker grid either with or without smoothed borders.

Waves For `Wav`, randomized volumetric damped cosine waves are added around the positions instead. The marker grid value m at point \mathbf{p} for a single wave around a center \mathbf{c} is defined as

$$m(\mathbf{p}) = \cos(w * \tilde{p}) * e^{-(3.7\tilde{p}/r)} \text{ where } \tilde{p} = \|\mathbf{p} - \mathbf{c}\|_2.$$

Here, r is the radius given by the randomized size that is computed as for `Sha`, and $w \sim \mathcal{U}(0.1, 0.3)$ is a randomized

waviness value, that determines the frequency of the damped wave.

B.4 Collected Data

The collected data sets `Iso`, `Cha`, `Mhd`, and `Tra` are based on different subsets from the Johns Hopkins Turbulence Database JHTDB (Perlman et al. 2007), that contain different types of data from direct numerical simulations (DNS). In these simulations, all spatial scales of turbulence up to the dissipative regime are resolved. The data set `SF` is based on the ScalarFlow data (Eckert, Um, and Thuerey 2019), that contains dense 3D velocity fields of real buoyant smoke plumes, created via multi-view reconstruction technique.

JHTDB The JHTDB subsets typically contain a single simulation with a very high spatial and temporal resolution and a variety of fields. We focus on the velocity fields, since turbulent flow data is especially complex and potentially benefits most from a better similarity assessment. We can mainly rely on using temporal sequences, and only need to add spatial jitters in some cases to increase the difficulty. As turbulence generally features structures of interest across all length scales, we create sequences of different spatial scales for each subset. To achieve this, we randomly pick a cutout scale factor s . If $s = 1$, we directly use the native spatial discretization provided by the database. For $s > 1$ we stride the spatial query points of the normal cubical cutout by s after filtering the data. For $s < 1$ the size of the cubical cutout is reduced by a factor of s in each dimension, and the cutout is interpolated to the full spatial size of 128^3 afterwards. Among other details, Tab. 4 shows the cutout scale factors, as well as the corresponding random weights.

ScalarFlow Since 100 reconstructions of different smoke plumes are provided in ScalarFlow, there is no need to add additional randomization to create multiple test sequences. Instead, we directly use each reconstruction sequence to create one similarity sequence in equal temporal steps. The only necessary pre-processing step is cutting off the bottom part of domain that contains the smoke inflow, since it is frequently not fully reconstructed. Afterwards, the data is interpolated to the full spatial size of 128^3 to match the other data sets.

B.5 Additional Example Sequences

Fig. 9, 10, and 11 show multiple full example sequences from all our data sets. In every sequence, the leftmost image is the baseline field. Moving further to the right, the change of one initial parameter increases for simulated data sets, and the spatio-temporal position offset increases for generated and collected data. To plot the sequences, the 3D data is projected along the z-axis to 2D via a simple mean operation. This means, noise that was added to the data or the simulation is typically significantly less obvious due to statistical averaging in the projection. Velocity data is directly mapped to RGB color channels, and scalar data is shown via different shades of gray. Unless note otherwise, the data is jointly normalized to $[0, 1]$ for all channels at the same time, via the overall minimum and maximum of the data field.

B.6 Data Distribution

There are many independent factors that influence the actual difficulty of the created data sequences. For example, our iterative data generation methods only calibrate Δ with the help of proxy functions. Furthermore, the clear correlation thresholds are only used for a small set of sequences, instead of sampling every sequence based on them.

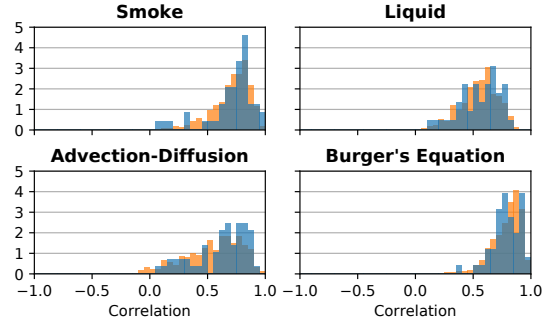


Figure 7: Normalized MSE correlation histograms of training (orange) and validation data (blue) with a bin size of 0.05, all of which roughly follow a truncated normal distribution.

As a result, the computed PCC values from the MSE (see Fig. 2) on the full data sets exhibit a natural, smooth distribution with controllable difficulty, instead of introducing an artificial distribution with hard cutoffs. According to the central limit theorem and taking into account that correlations have an upper bound of 1, we expect the PCC values to follow a truncated normal distribution. Intuitively, this corresponds to a distribution of trajectories with different curvature in the similarity model in Fig. 1. In fact, we empirically determined that only training data distributions in the PCC space which reasonably closely follow a truncated normal distribution with a sufficiently positive peak, result in a successful training of our model. We assume, that cutoffs or significantly different distributions indicate unwanted biases in the data, which prevented effective learning in our experiments. Fig. 7 shows normalized correlation histograms of our training and validation data sets, all of which roughly follow this desired truncated normal distribution.

C Training Stability for Loss Experiment

To further analyze the resulting stability of different variants of the loss investigated in Sec. 5, Fig. 8 shows training trajectories of multiple models. Displayed are the direct training loss over the training iterations, i.e. one loss value per batch, and a smoothed version via an exponential weighted moving average computed with $\alpha = 0.05$, corresponding to a window size of 40. The models are color-coded according to Fig. 4, and were trained with different batch sizes b , slicing values v , usage of the running sample mean RM , and usage of the correlation aggregation AG . Note that the models were generally trained with the same training seed, meaning the order of samples is very similar across all runs (only with potential deviations due to race conditions and GPU processing).

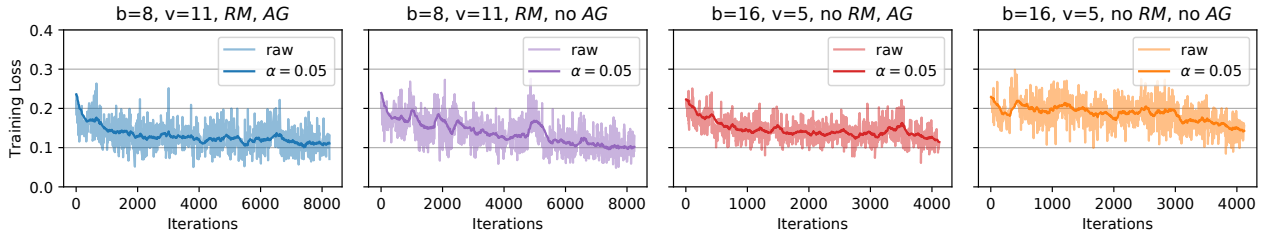


Figure 8: Loss curves from the loss function analysis in Fig. 4 for different models. Shown are the raw losses per batch over all training iterations (lighter line), and an exponential weighted moving average with $\alpha = 0.05$ (darker line) for better visual clarity.

In Fig. 8, it can be observed that models trained with lower batch sizes achieve lower training losses, leading to the better generalization on test and validation sets in Fig. 4. Using no *AG* leads to a less stable training procedure and a higher loss during most of training, especially for lower batch sizes. Occasionally, even large loss spikes occur that eradicate most training progress.

D Ablation Study Details

In the following, details for the ablation study models in Sec. 6 are provided. The proposed *VolSiM* metric uses the multiscale architecture *MS* described in Sec. 5 as a feature extractor. For all models, the general training setup mentioned in App. A stays identical, apart from: 1) removing the contribution of the entropy-based similarity model, 2) changes to the feature extractor, 3) a different feature extractor, 4) the training amount, or 5) the training data.

No Similarity Model The MS_{identity} model does not directly make use of the similarity model based on entropy. For that, the logarithmic transformation of $w_s = (j - i)/n$ according to the similarity model is replaced with an identity transformation during training. This corresponds to linear ground truth distances according to the order of each sequence determined by our data generation. Note that this simplified network version is slightly more efficient as the computation of the parameter c can be omitted. However, pre-computing these values for all sequences is computationally very light compared to training the full metric network.

Changes to *MS* Architecture For $MS_{3 \text{ scales}}$, the last scale block is removed, meaning the architecture from Fig. 3 ends with the $1/4$ resolution level, while the $1/8$ resolution level is omitted. For the $MS_{5 \text{ scales}}$ model, one additional scale block is added. In Fig. 3, this corresponds to a $1/16$ resolution level with a $16 \times 16 \times 16$ AvgPool and a scale block with 64 channels.

Simple CNN Feature Extractor The CNN_{trained} model employs an entirely different feature extractor network, similar to the classical AlexNet architecture (Krizhevsky, Sutskever, and Hinton 2017). It consists of 5 layers, each with a 3D convolution followed by a ReLU activation. The kernel sizes (12 - 5 - 3 - 3 - 3) decrease along with the strides (4 - 1 - 1 - 1 - 1), while the number of features first increases, then decreases again (32 - 96 - 192 - 128 - 128). To create some spatial reductions, two $4 \times 4 \times 4$ MaxPools with stride 2 are

included before the second and third convolution. The normalization and aggregation of the resulting feature maps is performed as described in App. A, in the same way as for the proposed feature extractor network. As a result, CNN_{trained} can also be applied in a fully convolutional manner for the scaling invariance experiment in Fig. 5.

Random Models For the random models CNN_{rand} and MS_{rand} , the corresponding feature extractor along with the aggregation weights are only initialized and not trained. However, to normalize every feature to a standard normal distribution, the training data is processed once to determine the feature mean and standard deviation before evaluating the models, as detailed in App. A.

Different Training Data For the $MS_{\text{added ISO}}$ model, we created 400 additional sequences from the JHTDB according to the *ISO* column in Tab. 4, that are added to the other training data. Note that they utilize different random seeds than the 60 test sequences, so the *ISO* data set becomes an additional validation set in Tab. 1. Similarly for the $MS_{\text{only ISO}}$ model, 1000 sequences according to the *ISO* column in Tab. 4 were collected. They replace all other training data from the original model, meaning the *ISO* data set becomes a validation set, and *Adv*, *Bur*, *Liq*, and *SmO* become further test sets in Tab. 1. For consistency, the *All* column in Tab. 1 still reports the combined values of the original test sets for both cases.

E Additional Ablations

To further investigate our method, we perform three additional ablations in the following. The first ablation focuses on the benefits of the resolution skip connections in our multiscale feature extractor. The other two replace the multiscale aspect in different ways, to demonstrate the robustness and memory efficiency of the proposed architecture. Additionally, directly using Pearson’s distance on our data is compared to an analysis using the MSE. Tab. 2 shows the resulting SRCC values for these models and the corresponding baselines on our data sets, which are computed as for Tab. 1.

No Skip Connections We compare the *VolSiM* model as proposed in the main paper with $MS_{\text{no skip}}$ that does not make use of the resolution skip connections (red dashed arrows in Fig. 3). This means each resolution is isolated, and sharing information across features of different levels is more difficult, even though all features from each layer are still used for the final distance computation. The results in the first block

Metric	Validation data sets				Test data sets										
	Adv	Bur	Liq	Smo	AdvD	LiqN	Sha	Wav	Iso	Cha	Mhd	Tra	SF	All	
<i>VolSiM</i>	0.75	0.73	0.66	0.77	0.84	0.88	0.95	0.96	0.77	0.86	0.81	0.88	0.95	0.85	
<i>MS_{no skip}</i>	0.80	0.70	0.78	0.75	0.86	0.88	0.80	0.95	0.76	0.86	0.78	0.86	0.91	0.82	
<i>MS_{4 scales}</i>	0.74	0.74	0.86	0.77	0.82	0.83	0.93	0.97	0.77	0.88	0.82	0.89	0.95	0.84	
<i>MS_{1 scale}</i>	0.73	0.71	0.78	0.75	0.83	0.76	0.91	0.96	0.73	0.87	0.78	0.88	0.94	0.82	
<i>MS_{with pool}</i>	0.70	0.70	0.76	0.72	0.82	0.79	0.94	0.96	0.76	0.83	0.79	0.88	0.97	0.83	
<i>MS_{no pool}</i>	0.73	0.70	0.72	0.71	0.83	0.77	0.94	0.95	0.76	0.87	0.78	0.86	0.90	0.82	
<i>MSE</i>	0.61	0.70	0.51	0.68	0.77	0.76	0.75	0.65	0.76	0.86	0.80	0.79	0.79	0.70	
<i>PCC</i>	0.65	0.69	0.55	0.65	0.72	0.80	0.72	0.73	0.70	0.83	0.75	0.78	0.89	0.69	

Table 2: Performance comparison of further ablation study models via the SRCC. Shown are a model where the resolution skip connections in the feature extractor are omitted (top block), and two models without a multiscale architecture (second and third block). The upper row in each block is the corresponding baseline. Furthermore, using Pearson’s distance directly as a metric is compared to an MSE evaluation (bottom block).

in Tab. 2 show that removing skip connections simplifies learning to some degree as the performance clearly increases on two validation sets. However, the generalization to data that is very different compared to the training distribution becomes more difficult. This is especially obvious for the data sets Sha and SF, both of which feature relatively large visual changes. Interestingly, the effects on the Wav data set with similar characteristics are only minor.

Singlescale Model To investigate the multiscale aspect of our architecture, we removed all scale blocks at lower resolutions, such that only the component at the native input resolution remains (see Fig. 3). To compensate for the lower number of network parameters, we scale the number channels in each layer by a factor of 11, leading to the rather shallow but wide network *MS_{1 scale}* with around 360k weights. The original network structure *MS_{4 scales}* with the four scale blocks reconstructs the ground truth more accurately for almost all data sets as displayed in the middle block of Tab. 2. Furthermore, the proposed structure requires about five times less memory during training.

No Pooling Layers A different approach to analyze the multiscale aspect, is to eliminate all AvgPool layers and set all convolution strides to 1 (also see Fig. 3). Here, no further adjustments to the network are required as neither pooling layers nor strides alter the number of network weights. The resulting *MS_{no pool}* model is only slightly worse compared to the baseline, as indicated by bottom block in Tab. 2. However, the *MS_{with pool}* model needs about ten times less memory compared to the adjust variant during training due to the significantly smaller feature sizes.

Note that for both ablations on the multiscale structure, the training and test data resolution was reduced to 32^3 . Furthermore, the number of channels in each layer for the *MS_{with pool}* and *MS_{no pool}* models was reduced by a factor of 0.5. Both choices are motivated by memory limitations and are the reason for the slightly different results across the three baseline models in Tab. 2.

Further Element-wise Metrics The last block in Tab. 2 analyzes the performance of Pearson’s distance on our data

sets. Here, d is computed via $d_i = 1 - |\text{PCC}(s_0, s_i)|$ and compared it to g via the SRCC in the same way as for the other metrics. The resulting performance is overall quite similar to an MSE evaluation, which is expected as Pearson’s distance essentially still is an element-wise comparison, even though it does take some general data statistics into account.

F Turbulence Case Study Details

For the case study in Sec. 6, the velocity field from the *isotropic1024coarse* data set from the JHTDB (Perlman et al. 2007) is utilized. Instead of calibrating sequences according to data generation approach and the similarity model, the data is directly converted to three sequences of different time spans without any randomization in this case. A spatial stride of 8 is employed for all cases, and to reduce memory consumption an additional temporal stride of 20 is used for the long time span only. The resulting sequences exhibit a spatial resolution of 128^3 , with 40 frames for the short span, 400 frames for medium span, and 200 frames for the long span. Before further processing, each frame is individually normalized to $[-1, 1]$. To create the results in Fig. 6, the first simulation frame t_1 is individually compared to all following simulation frames t_i via Pearson’s distance (red dashed trajectory) and *VolSiM* (blue solid trajectory). Note that *VolSiM* is applied fully convolutionally as it was trained on 64^3 data, and has to generalize to 128^3 here. For both cases, the resulting distances are normalized to $[0, 1]$ to visually compare them more easily. The examples from the sequences are visualized as described in App. B, while projecting along the x-axis.

References

- Eckert, M.-L.; Um, K.; and Thuerey, N. 2019. ScalarFlow: A Large-Scale Volumetric Data Set of Real-World Scalar Transport Flows for Computer Animation and Machine Learning. *ACM Transactions on Graphics*, 38(6).
- Holl, P.; Thuerey, N.; and Koltun, V. 2020. Learning to Control PDEs with Differentiable Physics. In *8th International Conference on Learning Representations (ICLR 2020)*. OpenReview.net.

- Kohl, G.; Um, K.; and Thuerey, N. 2020. Learning Similarity Metrics for Numerical Simulations. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, volume 119, 5349–5360.
- Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2017. ImageNet Classification with Deep Convolutional Neural Networks. *Communications of the ACM*, 60(6): 84–90.
- Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035.
- Perlman, E.; Burns, R.; Li, Y.; and Meneveau, C. 2007. Data Exploration of Turbulence Simulations Using a Database Cluster. In *Proceedings of the ACM/IEEE Conference on High Performance Networking and Computing*, 1–11.
- Thuerey, N.; and Pfaff, T. 2018. MantaFlow.
- Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 586–595.
- Zhu, Y.; and Bridson, R. 2005. Animating Sand As a Fluid. In *ACM Transactions on Graphics*, 965–972.

	Adv	Bur	Liq	Smo	AdvD	LiqN	Sha	Wav
Sequences train-val-test	398-57-0	408-51-0	405-45-0	432-48-0	0-0-57	0-0-30	0-0-60	0-0-60
Equation	Eq. 6	Eq. 7	Eq. 12, 13	Eq. 12, 13	Eq. 6	Eq. 12, 13	—	—
Simulator	PhiFlow ^d	PhiFlow ^d	MantaFlow ^e	MantaFlow ^e	PhiFlow ^d	MantaFlow ^e	MantaFlow ^e	MantaFlow ^e
Simulation setup	layered sines	layered sines	breaking dam + drop	rising plume with force field	layered sines	breaking dam + drop	random shapes	random damped waves
Time steps	120	120	80	120	120	80	—	—
Varied aspects	$f_1, f_2, f_3, f_4, f_5, f_7, o_1, o_2, o_d, noise$	$f_1, f_2, f_3, f_4, f_5, f_7, o_1, o_2, noise$	$drop_x, drop_y, drop_z, drop_{rad}, grav_x, grav_y, grav_z, noise$	$buoy_x, buoy_y, ff_{rot x}, ff_{rot z}, ff_{str x}, ff_{str z}, ff_{pos x}, ff_{pos y}, ff_{rad}, source_x, source_y, noise$	$f_1, f_2, f_3, f_4, f_5, f_7, o_1, o_2, o_d, noise$	$drop_x, drop_y, drop_z, drop_{rad}, grav_x, grav_y, grav_z, noise$	shape position	wave position
Noise integration	added to velocity	added to velocity	added to velocity	added to velocity	added to density	overlay on non-liquid	overlay on marker	overlay on marker
Used fields	density	velocity	velocity flags levelset	density pressure velocity	density	velocity	marker	marker

^d PhiFlow (<https://github.com/tum-pbs/PhiFlow>) from Holl, Thuerey, and Koltun (2020) ^e MantaFlow (<http://mantaflow.com/>) from Thuerey and Pfaff (2018)

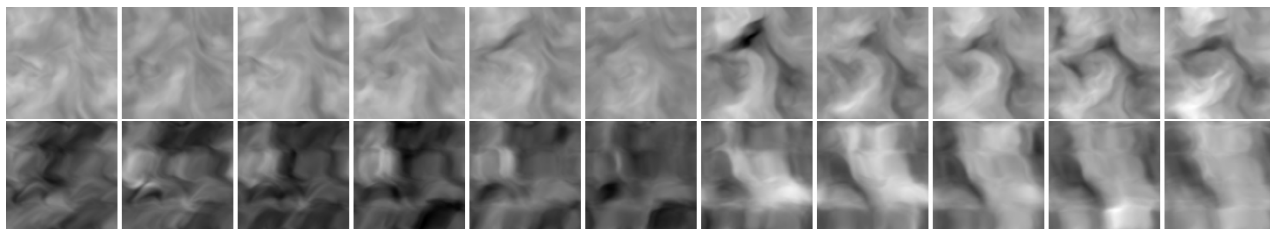
Table 3: Data set detail summary for the simulated and generated data sets.

	Iso	Cha	Mhd	Tra	SF
Sequences train-val-test	0-0-60	0-0-60	0-0-60	0-0-60	0-0-100
Repository	JHTDB – isotropic 1024coarse ^f	JHTDB – channel ^f	JHTDB – mhd1024 ^f	JHTDB – transition.bl ^f	ScalarFlow ^g
Repository size ^h $s \times t \times x \times y \times z$	$1 \times 5028 \times 1024 \times 1024 \times 1024$	$1 \times 4000 \times 2048 \times 512 \times 1536$	$1 \times 1024 \times 1024 \times 1024$	$1 \times 4701 \times 10240 \times 1536 \times 2048$	$100 \times 150 \times 100 \times 178 \times 100$ ⁱ
Temporal offset Δ_t	180	37	95	25	13
Spatial offset $\Delta_{x,y,z}$	0	0	0	0	0
Spatial jitter	0	0	25	0	0
Cutout scales	0.25, 0.5, 0.75, 1, 2, 3, 4	0.25, 0.5, 0.75, 1, 2, 3, 4	0.25, 0.5, 0.75, 1, 2, 3, 4	0.25, 0.5, 0.75, 1, 2	1
Cutout scale random weights	0.14, 0.14, 0.14, 0.16, 0.14, 0.14, 0.14	0.14, 0.14, 0.14, 0.16, 0.14, 0.14, 0.14	0.14, 0.14, 0.14, 0.16, 0.14, 0.14, 0.14	0.14, 0.14, 0.14, 0.30, 0.28	1
Used fields	velocity	velocity	velocity	velocity	velocity

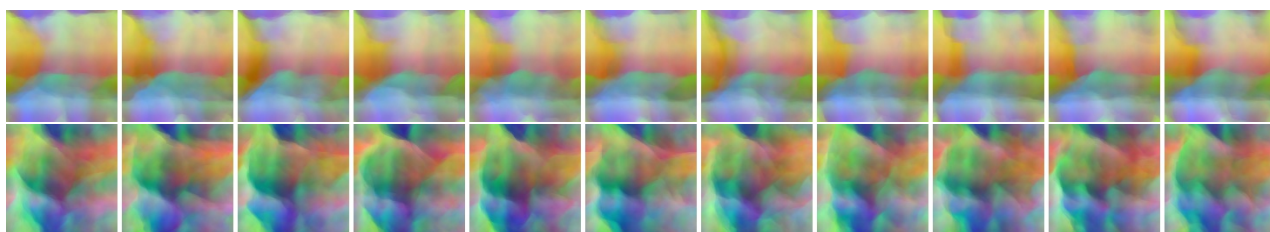
^f JHTDB (<http://turbulence.pha.jhu.edu/>) from Perlman et al. (2007) ^g ScalarFlow (<https://mediatum.ub.tum.de/1521788>) from Eckert, Um, and Thuerey (2019) ^h simulations $s \times$ time steps $t \times$ spatial dimensions x, y, z ⁱ cut to $100 \times 150 \times 100 \times 160 \times 100$ (removing 18 bottom values from y), since the smoke inflow is not fully reconstructed

Table 4: Data set detail summary for collected data sets.

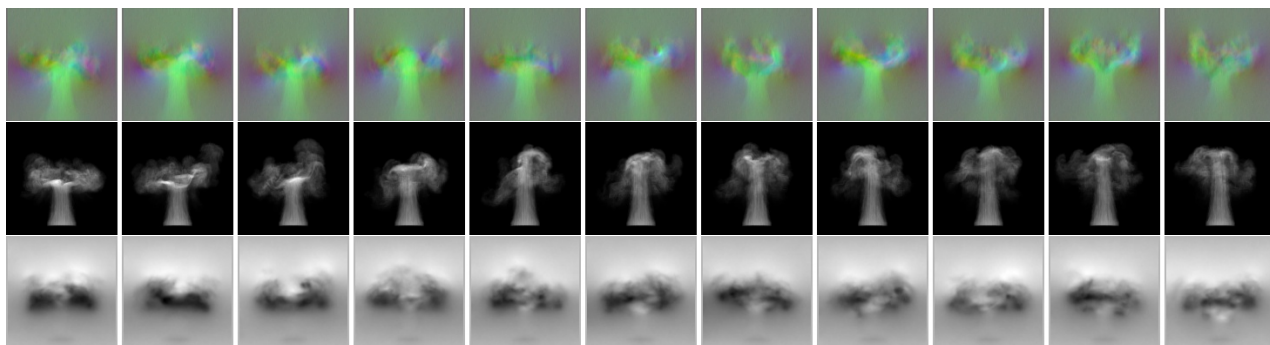
Adv: Advection-Diffusion ($2\times$ density)



Bur: Burgers' Equation ($2\times$ velocity)



Smo: Smoke (velocity, density, and pressure)



Liq: Liquid (velocity, levelset, and flags)

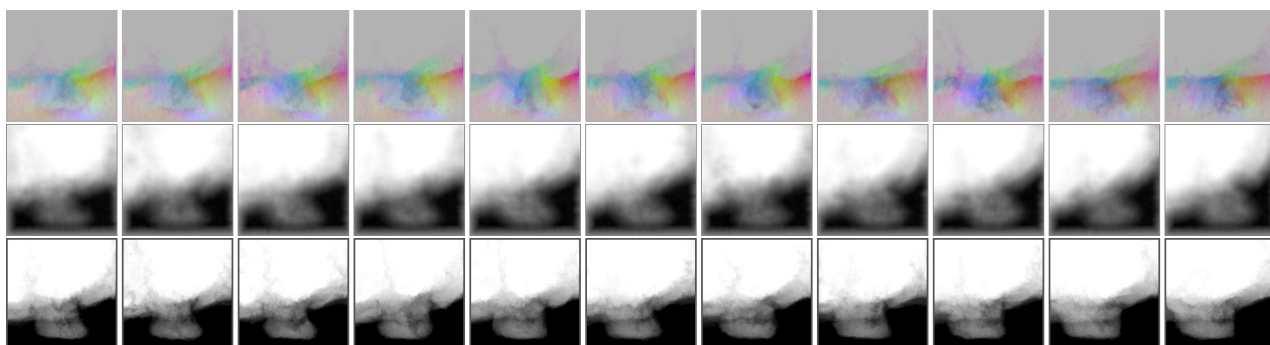
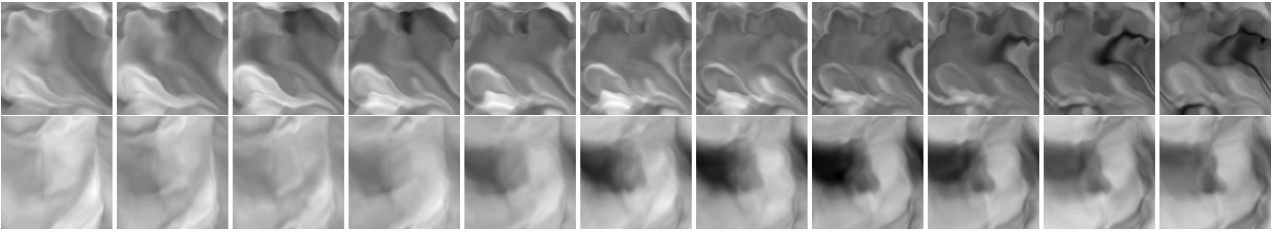
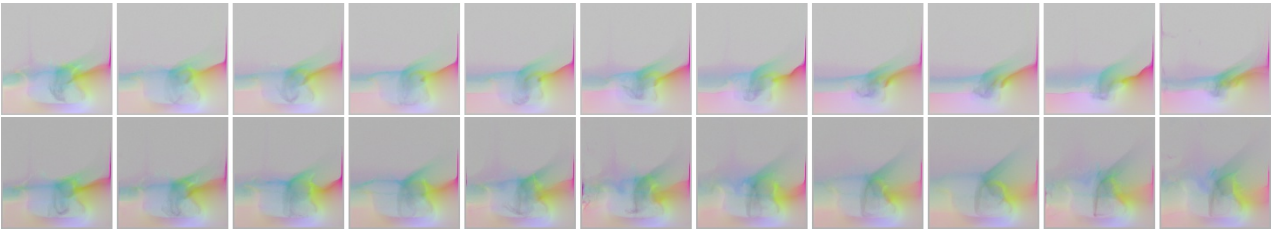


Figure 9: Example sequences of simulated training data, where each row features a full sequence from a different random seed.

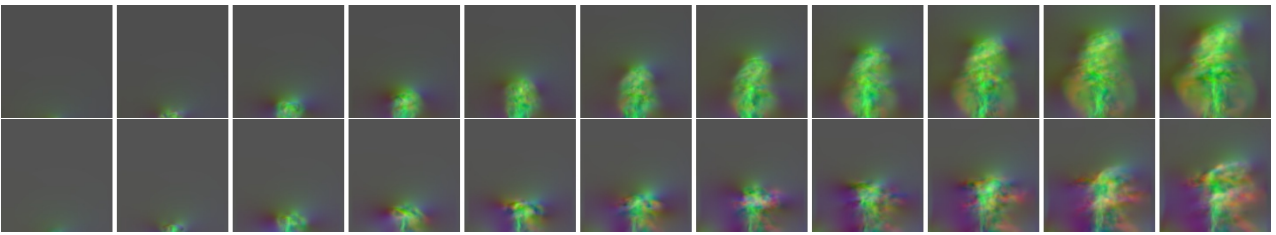
AdvD: Advection-Diffusion with density noise ($2\times$ density)



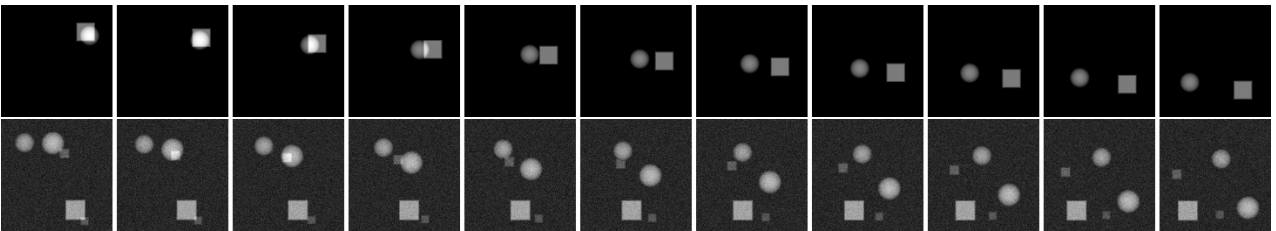
LiqN: Liquid with background noise ($2\times$ velocity)



SF: ScalarFlow ($2\times$ velocity)



Sha: Shapes ($2\times$ marker, without and with noise)



Wav: Waves ($2\times$ marker, without and with noise)

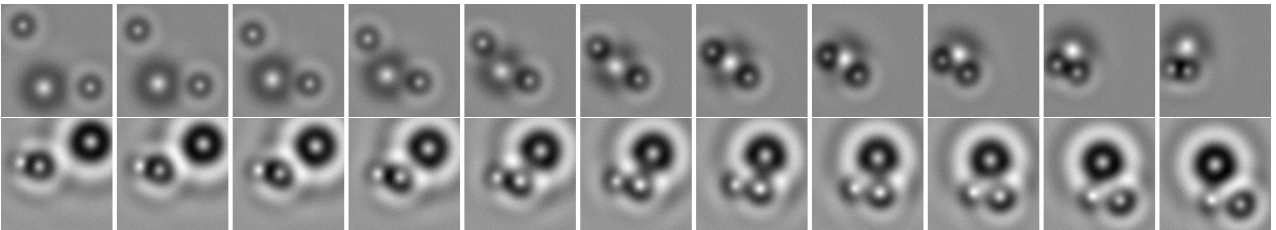
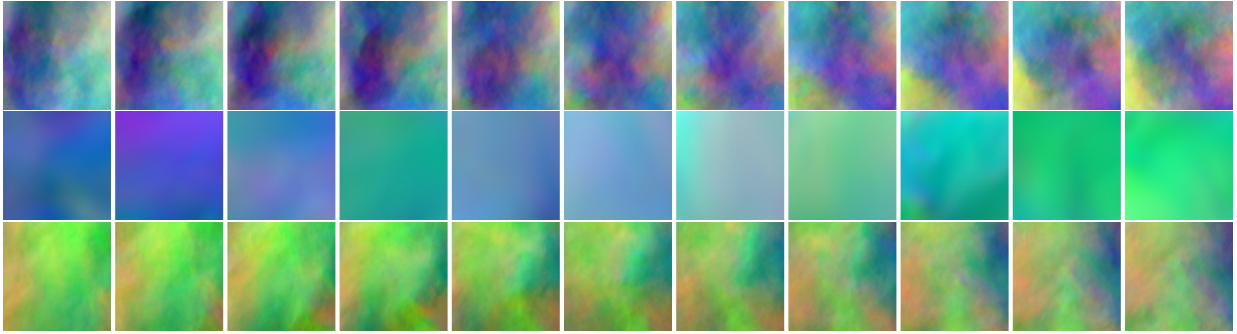
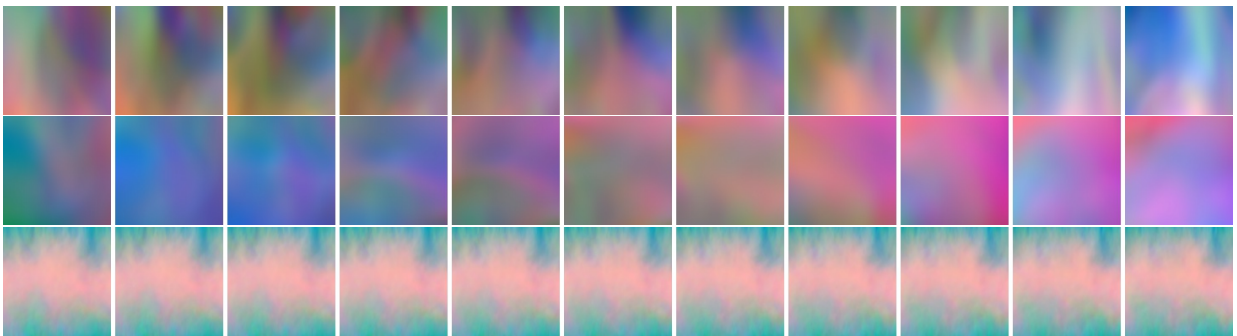


Figure 10: Example sequences of simulated (top two data sets), collected (middle data set), and generated (bottom two data sets) test data. Each row contains a full sequence from a different random seed. It is difficult to visually observe the background noise in LiqN due the projection along the z-axis to 2D, and due to image compression.

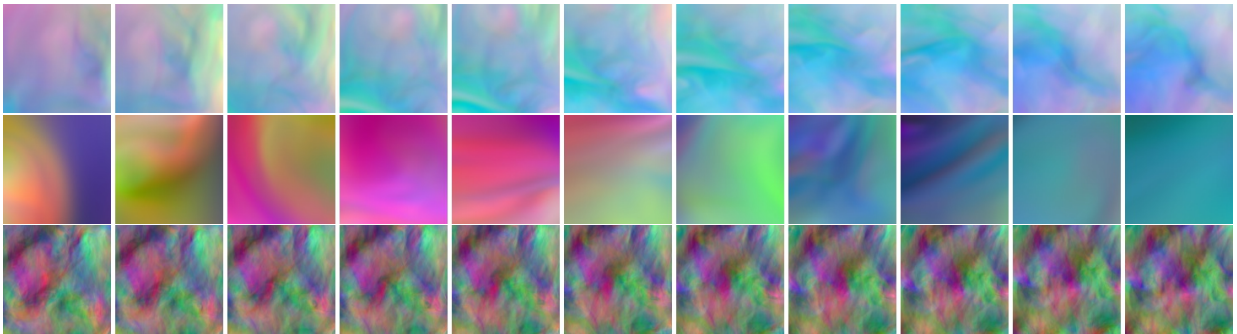
Iso: Isotropic turbulence ($3\times$ velocity)



Cha: Channel flow ($3\times$ velocity)



Mhd: Magneto-hydrodynamic turbulence ($3\times$ velocity)



Tra: Transitional boundary layer ($3\times$ velocity)

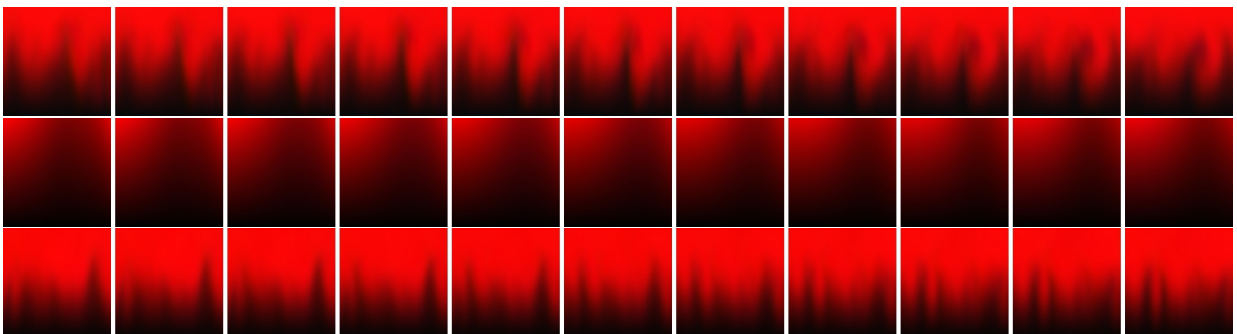


Figure 11: Example sequences of collected test data from JHTDB, where each row shows a full sequence from a different random seed. Notice the smaller cutout scale factor s for the middle example in each case. The predominant x-component in Cha is separately normalized for a more clear visualization.



Association for the Advancement of Artificial Intelligence

1900 Embarcadero Road, Suite 101, Palo Alto, California 94303

Palo Alto, California 94303 USA

AAAI COPYRIGHT FORM

Title of Article/Paper: Learning Similarity Metrics for Volumetric Simulations with Multiscale CNNs

Publication in Which Article/Paper Is to Appear: Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI-23)

Author's Name(s): Georg Kohl, Li-Wei Chen, Nils Thuerey

Please type or print your name(s) as you wish it (them) to appear in print

PART A – COPYRIGHT TRANSFER FORM

The undersigned, desiring to publish the above article/paper in a publication of the Association for the Advancement of Artificial Intelligence, (AAAI), hereby transfer their copyrights in the above article/paper to the Association for the Advancement of Artificial Intelligence (AAAI), in order to deal with future requests for reprints, translations, anthologies, reproductions, excerpts, and other publications.

This grant will include, without limitation, the entire copyright in the article/paper in all countries of the world, including all renewals, extensions, and reversions thereof, whether such rights current exist or hereafter come into effect, and also the exclusive right to create electronic versions of the article/paper, to the extent that such right is not subsumed under copyright.

The undersigned warrants that they are the sole author and owner of the copyright in the above article/paper, except for those portions shown to be in quotations; that the article/paper is original throughout; and that the undersigned right to make the grants set forth above is complete and unencumbered.

If anyone brings any claim or action alleging facts that, if true, constitute a breach of any of the foregoing warranties, the undersigned will hold harmless and indemnify AAAI, their grantees, their licensees, and their distributors against any liability, whether under judgment, decree, or compromise, and any legal fees and expenses arising out of that claim or actions, and the undersigned will cooperate fully in any defense AAAI may make to such claim or action. Moreover, the undersigned agrees to cooperate in any claim or other action seeking to protect or enforce any right the undersigned has granted to AAAI in the article/paper. If any such claim or action fails because of facts that constitute a breach of any of the foregoing warranties, the undersigned agrees to reimburse whomever brings such claim or action for expenses and attorneys' fees incurred therein.

Returned Rights

In return for these rights, AAAI hereby grants to the above author(s), and the employer(s) for whom the work was performed, royalty-free permission to:

1. Retain all proprietary rights other than copyright (such as patent rights).
2. Personal reuse of all or portions of the above article/paper in other works of their own authorship. This does not include granting third-party requests for reprinting, republishing, or other types of reuse. AAAI must handle all such third-party requests.
3. Reproduce, or have reproduced, the above article/paper for the author's personal use, or for company use provided that AAAI copyright and the source are indicated, and that the copies are not used in a way that implies AAAI endorsement of a product or service of an employer, and that the copies per se are not offered for sale. The foregoing right shall not permit the posting of the article/paper in electronic or digital form on any computer network, except by the author or the author's employer, and then only on the author's or the employer's own web page or ftp site. Such web page or ftp site, in addition to the aforementioned requirements of this Paragraph, shall not post other AAAI copyrighted materials not of the author's or the employer's creation (including tables of contents with links to other papers) without AAAI's written permission.
4. Make limited distribution of all or portions of the above article/paper prior to publication.
5. In the case of work performed under a U.S. Government contract or grant, AAAI recognized that the U.S. Government has royalty-free permission to reproduce all or portions of the above Work, and to authorize others to do so, for official U.S. Government purposes only, if the contract or grant so requires.

In the event the above article/paper is not accepted and published by AAAI, or is withdrawn by the author(s) before acceptance by AAAI, this agreement becomes null and void.

(1)

[Redacted Signature]

Author/Authorized Agent for Joint Author's Signature

03/12/2023

Date (MM/DD/YYYY)

Technical University of Munich

Employer for whom work was performed

Title (if not author)

(For jointly authored Works, all joint authors should sign unless one of the authors has been duly authorized to act as agent for the others.)

Association for the Advancement of Artificial Intelligence

1900 Embarcadero Road, Suite 101, Palo Alto, California 94303

Palo Alto, California 94303 USA

PART B – U.S. GOVERNMENT EMPLOYEE CERTIFICATION

This will certify that all authors of the above article/paper are employees of the U.S. Government and performed this work as part of their employment, and that the article/paper is therefore not subject to U.S. copyright protection. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations.

(2)

U.S. Government Employee Authorized Signature

Date (MM/DD/YYYY)

Name of Government Organization

Title (if not author)

(Please read and sign and return Part B only if you are a government employee and created your article/paper as part of your employment. If your work was performed under Government contract, but you are not a Government employee, sign only at signature line (1) in Part A and see item 5 under returned rights. Authors who are U.S. government employees should also sign signature line (1) in Part A above to enable AAAI to claim and protect its copyright in international jurisdictions.)

PART C—CROWN COPYRIGHT CERTIFICATION

This will certify that all authors of the above article/paper are employees of the British or British Commonwealth Government and prepared the Work in connection with their official duties, and that the article/paper is therefore subject to Crown Copyright and is not assigned to AAAI as set forth in the first sentence of the Copyright Transfer Section in Part A. The undersigned warrants that they are the sole author/translator of the above article/paper, and that the article/paper is original throughout, except for those portions shown to be in quotations, and acknowledges that AAAI has the right to publish, distribute, and reprint the Work in all forms and all media.

(3)

British or British Commonwealth Government Employee Authorized Signature

Date (MM/DD/YYYY)

Name of Government Organization

Title (if not author)

(Please read and sign and return Part C only if you are a British or British Commonwealth Government employee and the Work is subject to Crown Copyright. Authors who are British or British Commonwealth government employees should also sign signature line (1) in Part A above to indicate their acceptance of all terms other than the copyright transfer.)