Learning Similarity Metrics for Numerical Simulations

Georg Kohl [1] **Kiwon Um** [1] **Nils Thuerey** [1]

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 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

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

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

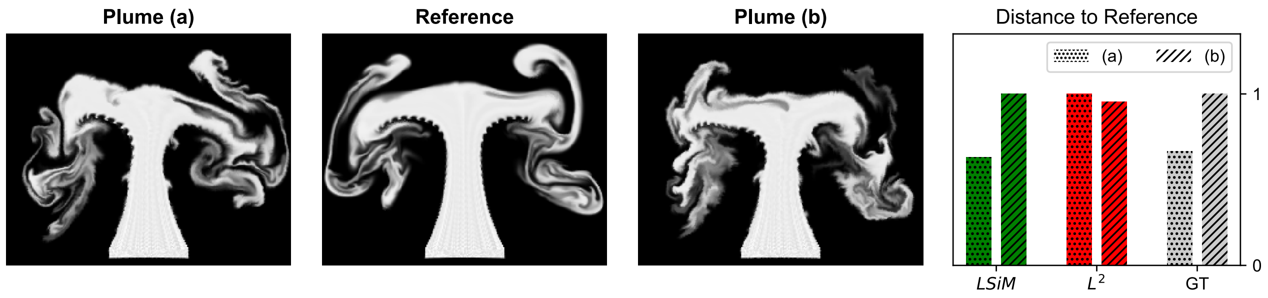


Figure 1. Example of field data from a fluid simulation of hot smoke with normalized distances for different metrics. Our method (*LSiM*, green) approximates the ground truth distances (GT, gray) determined by the data generation method best, i.e., version (a) is closer to the ground truth data than (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 $m : \mathbb{I} \times \mathbb{I} \to [0, \infty)$ is a metric on its input space $\mathbb{I}$ if it satisfies the following properties $\forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathbb{I}$:

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

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

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

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

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 $m(\boldsymbol{x}, \boldsymbol{x}) = 0 \ \forall \boldsymbol{x} \in \mathbb{I}$, 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

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

computed by our network is split into two parts: $m_1 : \mathbb{I} \to \mathbb{L}$ to compute the latent space embeddings $\tilde{\boldsymbol{x}} = m_1(\boldsymbol{x}), \tilde{\boldsymbol{y}} = m_1(\boldsymbol{y})$ from each input, and $m_2 : \mathbb{L} \to [0, \infty)$ to compare these points in the latent space $\mathbb{L}$. We chose operations for m_2 such that it forms a metric $\forall \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}} \in \mathbb{L}$. 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 $m(\boldsymbol{x}, \boldsymbol{x}) = 0 \ \forall \boldsymbol{x} \in \mathbb{I}$ 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-

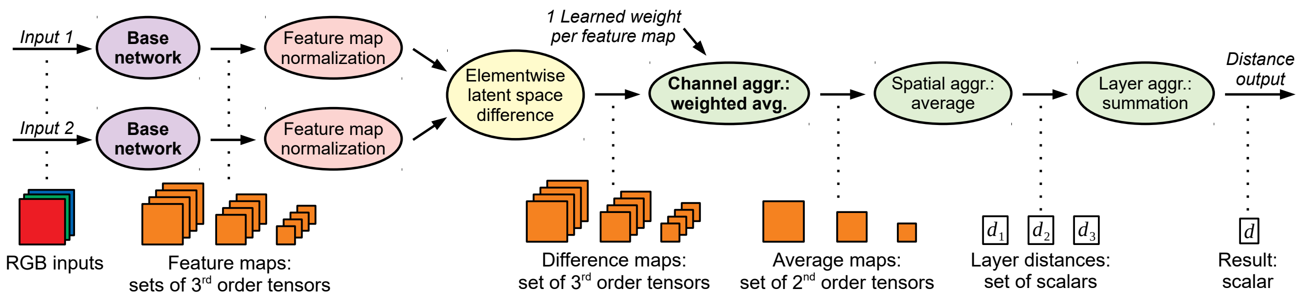

Input 1
Input 2
Base network
Base network
Feature map normalization
Feature map normalization
Elementwise latent space difference
1 Learned weight per feature map
Channel aggr.: weighted avg.
Spatial aggr.: average
Layer aggr.: summation
Distance output
RGB inputs
Feature maps: sets of 3rd order tensors
Difference maps: set of 3rd order tensors
Average maps: set of 2nd order tensors
Layer distances: set of scalars
Result: scalar
d1
d2
d3
d

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 (Thuerey 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

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

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:

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

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 expensive[1], 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 $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}} \in \mathbb{L}$ that consist of all extracted features from the two inputs $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{I}$ 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 $\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{y}}_i \;\forall i \in \{0, 1, \dots, \dim(\mathbb{L})\}$ is not suitable, as it invalidates non-negativity and symmetry. Instead, exponentiation of an absolute difference via $|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{y}}_i|^p$ yields an L^p metric on $\mathbb{L}$, when combined with the correct aggregation and a pth root. $|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{y}}_i|^2$ 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 $|\tilde{\boldsymbol{x}}_i - \tilde{\boldsymbol{y}}_i|$ 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.

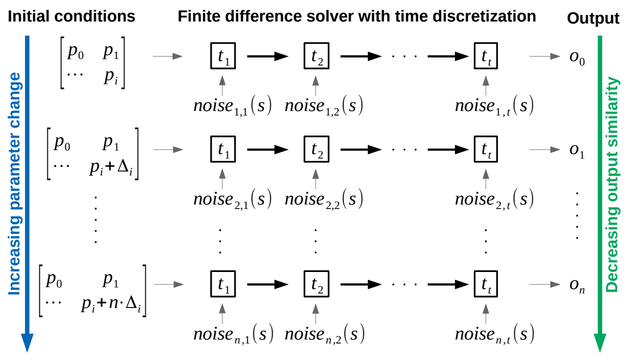

Initial conditions
Finite difference solver with time discretization
Output
Increasing parameter change
Decreasing output similarity
p_0 p_1 $\cdots$ p_i
p_0 p_1 $\cdots$ $p_i{+}\Delta_i$
p_0 p_1 $\cdots$ $p_i{+}n{\cdot}\Delta_i$
t_1 t_2 t_t
o_0 o_1 o_n
$noise_{1,1}(s)$ $noise_{1,2}(s)$ $noise_{1,t}(s)$
$noise_{2,1}(s)$ $noise_{2,2}(s)$ $noise_{2,t}(s)$
$noise_{n,1}(s)$ $noise_{n,2}(s)$ $noise_{n,t}(s)$

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 similarity 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 simulation 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 $\boldsymbol{c} \in [0, 1]^{0.5(n+1)n}$ representing the parameter change from the data generation. For a distance prediction $\boldsymbol{d} \in [0, \infty)^{0.5(n+1)n}$ of our network for one sample, we compute the loss with:

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

Here, the mean of a distance vector is denoted by $\bar{\boldsymbol{c}}$ and $\bar{\boldsymbol{d}}$ 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.

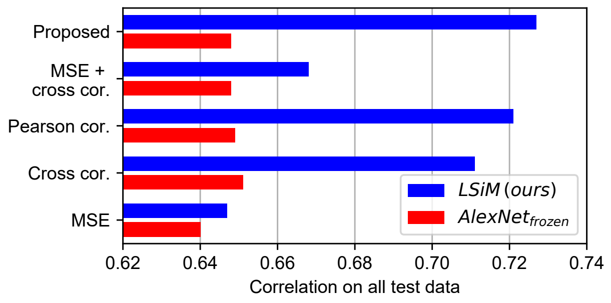


Figure 4. Performance comparison on our test data of the proposed approach (*LSiM*) and a smaller model (*AlexNet$_{frozen}$*) for different loss functions on the y-axis.

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 $(\boldsymbol{c} \cdot \boldsymbol{d}) \, / \, (\|\boldsymbol{c}\|_2 \, \|\boldsymbol{d}\|_2)$. 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.

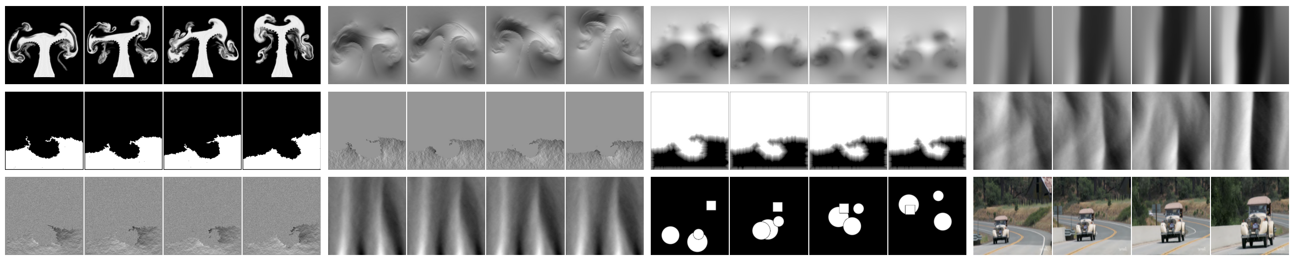
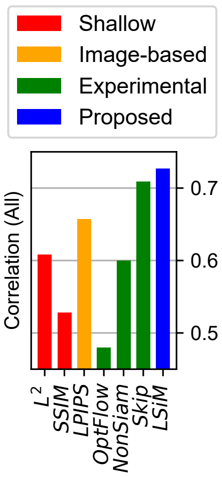


Figure 5. Samples from our data sets. For each subset the reference is on the left, and three variations in equal parameter steps follow. From left to right and top to bottom: `Smo` (density, velocity, and pressure), `Adv` (density), `Liq` (flags, velocity, and levelset), `Bur` (velocity), `LiqN` (velocity), `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	*0.58*	0.79	0.61
SSIM	0.69	0.73	0.77	0.71	0.77	*0.26*	**0.69**	*0.46*	0.75	*0.53*
LPIPS v0.1.	0.63	0.68	0.68	0.72	**0.86**	*0.50*	0.62	0.84	**0.83**	0.66
AlexNet$_{random}$	0.63	0.69	0.69	0.66	0.82	0.64	0.65	*0.67*	0.81	0.65
AlexNet$_{frozen}$	0.66	0.70	0.69	0.71	**0.85**	*0.40*	0.62	0.87	**0.84**	0.65
Optical flow	0.62	*0.57*	*0.36*	*0.37*	*0.55*	*0.49*	*0.28*	*0.61*	0.75	*0.48*
Non-Siamese	0.77	**0.85**	0.78	**0.74**	*0.65*	**0.81**	0.64	*0.25*	0.80	0.60
Skip$_{from\ scratch}$	**0.79**	0.83	**0.80**	**0.74**	**0.85**	0.78	0.61	0.78	**0.83**	0.71
LSiM$_{noiseless}$	0.77	0.77	0.76	0.72	**0.85**	0.62	0.58	0.86	0.82	0.68
LSiM$_{strong\ noise}$	0.65	*0.65*	0.67	0.69	0.84	*0.39*	0.54	**0.89**	0.82	0.64
LSiM (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*)

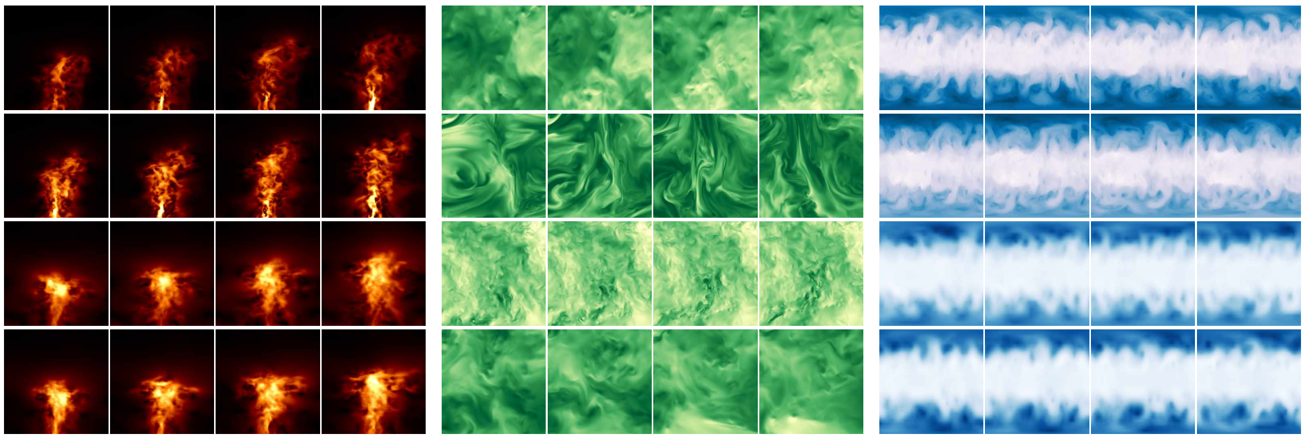


Figure 6. Examples from three real-world data repositories used for evaluation, visualized via color-mapping. Each block features four different sequences (rows) with frames in equal temporal or spatial intervals. Left: *ScalarFlow* – captured buoyant volumetric transport flows using the z-slice (top 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.

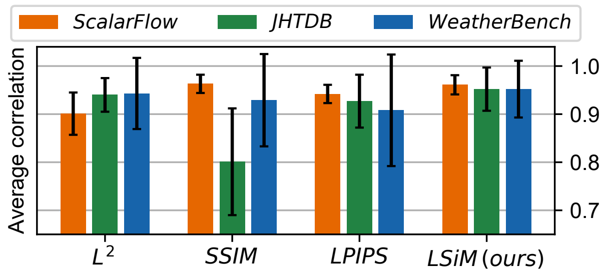


Figure 7. Spearman correlation values for multiple metrics on data from three repositories. Shown are mean and standard deviation over different temporal or spatial intervals used to create sequences.

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 geopotential 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 Sience 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., Hahm, 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.