informatics

MDPI

Article

Sampling and Estimation of Pairwise Similarity in Spatio-Temporal Data Based on Neural Networks

Steffen Frey

Visualization Research Institute, University of Stuttgart; 70569 Stuttgart, Germany; steffen.frey@visus.uni-stuttgart.de; Tel.: +49-(0)-711-6858-8629

Academic Editors: Achim Ebert and Gunther H. Weber
Received: 1 June 2017 ; Accepted: 18 August 2017; Published: 26 August 2017

Abstract: Increasingly fast computing systems for simulations and high-accuracy measurement techniques drive the generation of time-dependent volumetric data sets with high resolution in both time and space. To gain insights from this spatio-temporal data, the computation and direct visualization of pairwise distances between time steps not only supports interactive user exploration, but also drives automatic analysis techniques like the generation of a meaningful static overview visualization, the identification of rare events, or the visual analysis of recurrent processes. However, the computation of pairwise differences between all time steps is prohibitively expensive for large-scale data not only due to the significant cost of computing expressive distance between high-resolution spatial data, but in particular owing to the large number of distance computations ($O(|T|^2)$), with $|T|$ being the number of time steps). Addressing this issue, we present and evaluate different strategies for the progressive computation of similarity information in a time series, as well as an approach for estimating distance information that has not been determined so far. In particular, we investigate and analyze the utility of using neural networks for estimating pairwise distances. On this basis, our approach automatically determines the sampling strategy yielding the best result in combination with trained networks for estimation. We evaluate our approach with a variety of time-dependent 2D and 3D data from simulations and measurements as well as artificially generated data, and compare it against an alternative technique. Finally, we discuss prospects and limitations, and discuss different directions for improvement in future work.

Keywords: time-dependent data; neural networks; adaptive sampling; volume visualization

---

1. Introduction

Time-dependent data sets with increasing resolution in both time and space are generated at a fast rate, enabled by advances in parallel computing systems for simulations and high-accuracy measurement techniques. This data can feature millions of cells and thousands of time steps, and thus poses significant challenges for visual analysis. Even if the complete data—well-exceeding the available memory in most cases—could be presented to the user interactively, still numerous issues due to visual clutter and occlusion need to be prevented. A popular and natural choice to visualize the data without (temporal) occlusion and clutter issues is animation (i.e., sequentially rendering individual time steps). However , it has been shown to be ineffective as only a limited number of frames can be memorized by an observer (e.g., [1]). This motivates the development of visualization approaches that select and/or aggregate data in a data-driven way to enable efficient visual analysis and exploration.

For some of these type of data-driven visualization approaches, the computation of mutual distances between time steps is a fundamental operation for automatic analysis techniques. Recent examples of applications in visualization include the generation of a meaningful static overview visualization, the identification of rare events, as well as the visual analysis of recurrent processes.

Apart from that, visualizing the similarity information alone directly can also drive interactive visual exploration by indicating processes of interest to a user. In general, these application scenarios require the full computation of pairwise differences between all time steps. This is prohibitively expensive in particular in the context of large-scale data. This is not only due to (1) the significant cost of computing expressive distance between high-resolution time steps, but (2) especially owed to the large number of distance computations involved ($O(|T|^2)$), with $|T|$ being the number of time steps). This means that thousands of time steps already induce millions of (costly) distance computations.

In this work, we present and evaluate different strategies for the progressive computation of similarity information in a time series, as well as an approach for estimating missing distance information based on neural networks. Different strategies for sampling we consider range from purely random sampling over uniform (data-agnostic) to adaptive (data-driven) sampling strategies. With this sampled similarity information (i.e., a subset of a pairwise distances D of a time series T), we then aim to reconstruct the full set of pairwise distances D using a neural network. The goal of this approach is to let the neural network implicitly capture the special structure and properties of similarity information in spatio-temporal data. We then essentially combine both aspects by training neural networks for similarity estimation particularly on different sampling patterns of the different strategies. This eventually allows to automatically determines the sampling strategy yielding the best result in combination with trained networks for estimation.

The remainder of this paper is structured as follows. First, we review related work in Section 2. Then, we introduce our problem statement and give an overview of our approach in Section 3. We then discuss the two main parts of this work: different strategies for the sampling of similarity data (Section 4), as well as neural networks for the estimation of similarities (Section 5). Finally, we evaluate and discuss the properties of our approach in Section 6, before concluding our work in Section 7.

2. Related Work

2.1. Visualization of Spatio-Temporal Data

For the visualization of time-varying data, and extensions to many techniques could be applied to make them more general towards dealing with multi-field data. Lee and Shen [2] visualize trend relationships among variables in multivariate time-varying data. Joshi and Rheingans [3] evaluate illustration-inspired techniques for time-varying data, like speedlines or flow ribbons. One approach is to interpret the data as a space-time hypercube, and apply extended classic visualization operations like slicing and projection techniques [4] or temporal transfer functions [5] to it (cf. Bach et al. [6] for on overview). Another way to approach time-dependent data are feature-based techniques. Here, particularly Time Activity Curves (TAC) that contain each voxel's time series have been used as the basis for different techniques (e.g., [7]). Apart from such techniques working directly with scalar volume data, a large body of work in time-dependent volume visualization is based on feature extraction. Wang et al. [8] extract a feature histogram per volume block (typically hundreds to thousands of voxels). They then derive entropy-based importance curves that characterize the local temporal behavior of each block, and classify them via k-means clustering. Widanagamaachchi et al. [9] employ feature tracking graphs. Lee and Shen [10] visualize time-varying features and their motion on the basis of time activity curves (TAC) that contain each voxel's time series. Fang et al. [7] use TACs in combination with different similarity measures. Silver et al. [11] isolate and track representations of regions of interest. The robustness of this approach has been improved by Ji and Shen [12] with a global optimization correspondence algorithm based on the Earth Mover's Distance. Scale-space based methods and topological techniques have also been used here (e.g., [13,14]). Schneider et al. [15] compare scalar fields on the basis of the largest contours.

Another line of techniques is based on the direct comparison of time steps. The Earth Mover's Distance (EMD, also known as the Wasserstein metric) is a common metric to compute the difference between mass distributions (conceptually, it determines the minimum cost of turning one (mass) distribution into the other) [16]. For instance, Tong et al. [17] use different metrics to compute the distance between data sets, and employ dynamic programming to select the most interesting time steps accordingly. The field of video analysis also deals with related analysis problems, yet typically employing different methodologies. Specialized image and video metrics are used to compare frames (e.g., [18]), and distinct approaches were proposed to generate summaries of videos, e.g., based on the motion of actors over time [19]. In addition, illustrative techniques have been used to depict processes of interest. Lu and Shen [20] propose interactive storyboards composed of volume renderings and descriptive geometric primitives. While most techniques mentioned above deal with volume data, numerous approaches have been presented for flow visualization (cf. Post et al. [21] and McLoughlin et al. [22] for an overview). Note that different fields have developed different methodologies to quantify similarity for other application settings. For instance, to enable style-based search, Garces et al. [23] present a method for measuring the similarity in style between two pieces of vector art, based on the differences between four types of features: color, shading, texture, and stroke. Feature weightings are learned from crowdsourced experiments.

Frey and Ertl [24] presented a technique to generate transformations between arbitrary volumes, providing both expressive distances and smooth interpolates. On this basis, they presented a new approach for the streaming selection of time steps in temporal data that allows for the reconstruction of the full sequence with a user-specified error bound. An accelerated version with overall improved efficiency as well as an extension to manycore devices (i.e., GPUs) has been presented in a follow-up work [25]. We use this approach to quantify distances between time steps in this paper.

On the basis of similarity information between time steps, Frey and Ertl [26] adaptively select time steps from time-dependent volume data sets for an integrated and comprehensive visualization. This reduced set of time steps not only saves cost, but also allows to show both the spatial structure and temporal development in one combined rendering. The selection optimizes the coverage of the complete data on the basis of a minimum-cost flow-based technique to determine meaningful distances between time steps. An interactive volume raycaster produces an integrated rendering of the selected time steps, and their computed differences are visualized in a dedicated chart to provide additional temporal similarity information.

2.2. Similarity Matrices to Directly Visualize and Analyze Similarity Information

Their benefits and utility of recurrence plots and similarity matrices are discussed in detail by Marwan et al. [27]. There are also variants that extend those concepts from univariate to multivariate data. One possibility to apply these concepts to study the spatial structure of data is to separate the data into many one-dimensional data series, and to apply the recurrence analysis separately to each of these series [28]. Another possibility is the extension of the temporal approachof recurrence plots to a spatial one [29] at the cost of high-dimensional domains, e.g., a time-dependent 2D image is mapped to a 4D recurrence plot, which is, however, hard to visualize. Bautista et al. [30] analyze the difference between recurrence plots from different time series. For multi-field visualization, Frey et al. [31] presented an interactive approach on the basis of similarity matrices for extracting and exploring time-dependent phenomena, that allows to compare different locations, modalities, ensemble runs, or generally even data sets with no direct relation. It focuses on periodic and quasi-periodic behavior at single points, but was also used to analyze cross-correlations in ensemble and multi-variate data.

2.3. Machine Learning for Image Interpolation and Similarity Learning

Machine learning is popularly regarded as the only viable approach to building AI systems that can deal with (very) complicated environments [32]. In particular, in this work, we employ neural networks for estimating the similarity between different time steps. Image interpolation is a different task with different inherent characteristics, but there are also some related aspects. Hu et al. [33] propose an interpolation algorithm using a classification-based neural network approach with the goal to improve the image quality. Plaziac [34] compared two adaptive algorithms for image interpolation based on a multilayer perceptron. More recently, Chen et al. [35] used anisotropic probabilistic neural network on the basis of an anisotropic Gaussian kernel to provide high adaptivity of smoothness/sharpness during image/video interpolation.

Neural networks have also been applied to similarity learning, which belongs to the category of supervised machine learning in artificial intelligence. In general, this resembles our task of learning the similarities between time steps of spatio-temporal data, yet previous work has been done mostly for very different application scenarios. The goal is to learn from examples a similarity function that measures how similar or related two entities are. It has applications in ranking, in recommendation systems, visual identity tracking, face verification, and speaker verification. Guillaumin et al. [36] present two methods for learning robust distance measures for assessing the similarity of faces. Similarity learning is closely related to distance metric learning, in that metric learning is the task of learning a distance function over objects. Kulis [37] presents an overview of existing research in metric learning. Davis et al. [38] present an information-theoretic approach to learning a Mahalanobis distance function (the Mahalanobis distance is a measure of the distance between a point and a distribution).

3. Overview

The motivation behind this work is to gain insights from spatio-temporal data. This data can have different processes and structures, and be obtained via measurements and different types of CFD simulations (Figure 1). This means that this type of analysis is of interest to a wide variety of different fields. In this section, we first provide an introduction into similarity information from time series data in Section 3.1. We then cover the fundamentals of neural networks that are the basis for our similarity estimation approach discussed later in this work (Section 3.2). Finally, we give an outline of our approach and its different components in Section 3.3.

3.1. Similarity Information from Spatio-Temporal Data

We aim to analyze our time series data T on the basis of similarities between individual time steps $t \in T$. Here, the similarity between individual time steps is quantified by function $d : T \times T \to \mathbb{R}$, with the result being in the range $[0, \infty)$ (0 denotes identity). We further assume $d(\cdot, \cdot)$ to be symmetric, i.e., $d(t_0, t_1) = d(t_1, t_0)$ for $t_0, t_1 \in T$. In the following, we therefore only consider $d(t_0, t_1)$ for $t_0 < t_1$ (in the identity case of $t_0 = t_1$, $d(t_0, t_1) = 0$). As a basis for numerous analysis applications, in this work we are interested in obtaining all pair-wise similarities $d(t_0, t_1)$ between time steps $t_0, t_1 \in T$ with $t_0 < t_1$. This means that for $|T|$ time steps there are $|d(T)|$ (also denoted as $|D|$) time steps:

$$|d(T)| = \frac{(|T| - 1) \cdot |T|}{2} \tag{1}$$

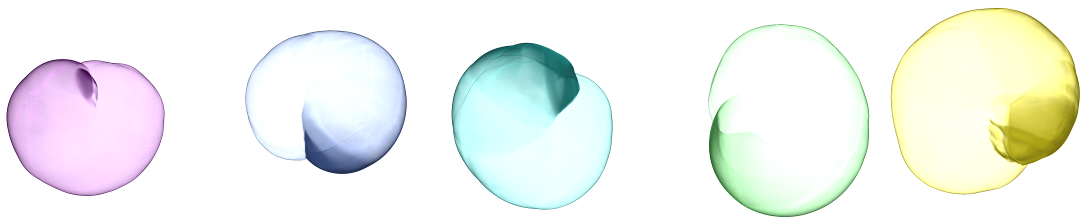
For the real-world data sets in Figure 1, the results are shown in Figure 2 in the form of similarity plots. We compute similarity information from the real-world data using the metric by Frey and Ertl [24,25] (cf. discussion later in Section 5.2). Essentially, it constitutes a fast computation method of the Earth Mover's Distance ([16], also known as the Wasserstein metric) that makes it computationally feasible to apply it directly to high-resolution data. While the computation of similarities between all pairs of time steps is very expensive, for training and testing purposes, we generated reference similarity information for the time series data introduced above over the course of several weeks on different machines (using both CPUs and GPUs). Most notably, due the symmetric property of the distance quantification function d as discussed above (i.e., $d(t_0, t_1) = d(t_1, t_0)$ for $t_0, t_1 \in T$), not a full pair-wise matrix is shown but a only the cases for $t_0 < t_1$. In the end, this forms a triangle-shaped plot. Different similarity structures can be seen, with the Kármán and most notably the Supernova data set featuring processes at distant points in time that are very similar (in the Kármán, this can be observed in the bottom right where line structures with a small offset to the diagonal can be seen).

(**a**) **Bottle** (resolution 900 × 430, 160 time steps considered): laser pulse shooting through a bottle, captured via Femto Photography (Velten et al. [39]).

(**b**) **von Kármán** (resolution 301 × 101, 418 time steps): 2D time-dependent CFD simulation of a von Kármán vortex street.

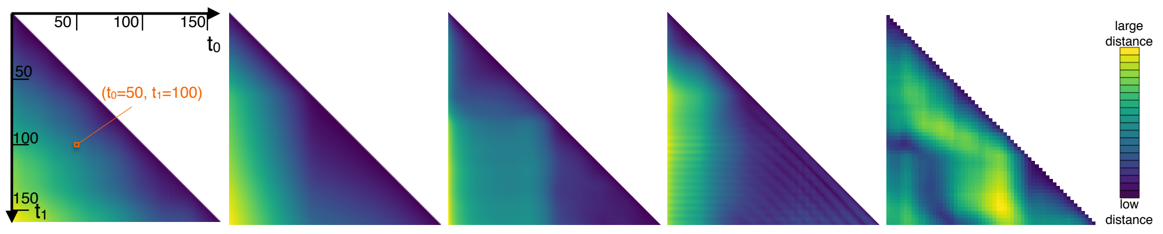
(**c**) **Hot Room** (resolution 101 × 101, 265 time steps): air flow within a closed container, driven by buoyant forces imposed by a heated bottom plate and a cooled top plate. To provoke transient aperiodic flow, the container exhibits two barriers (one on the top, one on the bottom).

(**d**) **Droplet** (256^3, 155 time steps): two drops colliding asymmetrically (courtesy of C. Meister, Institute of Aerospace Thermodynamics, University of Stuttgart).



(**e**) **Supernova** (432^3, 60 time steps): result of a supernova simulation. The data set is made available by Dr. John Blondin at the North Carolina State University through US Department of Energy's SciDAC Institute for Ultrascale Visualization.

Figure 1. All data sets include scalar values, that are mapped to a representation that is shown here, and also used for the distance computation via a user-defined transfer function (respective distances of each time series are plotted in Figure 1).



(**a**) Bottle. (**b**) Droplet. (**c**) Hot Room. (**d**) von Kármán. (**e**) Supernova.

Figure 2. Input similarity information from different data sets presented in the form of similarity matrices (with t_0 along the x-axis and t_1 along the y-axis, cf. (**a**)). Only one half of the matrix is visualized due the symmetry property of our distance metric (i.e., $d(t_0, t_1) = d(t_1, t_0)$). Values are mapped to colors using the viridis color map (low distances $\widehat{=}$ purple, medium $\widehat{=}$ green/blue, large $\widehat{=}$ yellow).

3.2. Neural Networks Basics (for Time Series Similarity Estimation)

In this work, we make use of so-called feedforward neural networks (aka multi- layer perceptrons (MLPs)) [32]. The goal of a feedforward network is to approximate some function f^*. In our case, we aim to estimate the result distance function $d(t_0, t_1)$ that determines the similarity between any two time steps in a time series $t_0, t_1 \in T$. In general, a feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best function approximation. In our application scenario, both input and out of the neural network is pair-wise similarity information in a time series.

These kind of models are called feedforward because information flows through the function being evaluated from x, through the intermediate operations used to define f, and eventually to the output f. There are no feedback connections in which outputs of the model are fed back into itself (as in recurrent neural networks, e.g., [32]).

Feedforward neural networks are typically represented by composing together different functions. The model is associated with a directed acyclic graph describing how the functions are composed together. For example, there might be three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}))$. These chain structures are the most commonly used structures of neural networks. In this case, $f^{(1)}$ is called the first layer of the network, $f^{(2)}$ is called the second layer, and so on. While the first layer is denote as the input layer, the final layer of a feedforward network is called the output layer. During neural network training, we adjust $f(x)$ to match $f^*(x)$. Each example x is accompanied by reference result values $y = f^*(x)$. The training examples specify directly what the output layer must do at each point x; it must produce a value that is close to y. As the training data does not show the desired output for the layers between the input and the output layer, these layers are called hidden layers.

3.3. Approach Outline

A conceptual overview on our approach is given in Figure 3. Essentially, there are two distinct phases, that are indicated by the wide gray arrows: (1) select and (2) optimize. First, in selection, we evaluate different sampling strategies to sample similarity information. Sampling strategies define different techniques to approach the adaptive sampling of similarity information (cf. Section 4). We then use the full similarity information along with the sparse set generated by the adaptive sampling for training a neural network to reconstruct the full information. We then validate the generated model, which essentially results in an error value (subsequently denoted as *cost*). Using this information, we compare the obtained cost values of all sampling strategies, and choose the strategy yielding the lowest cost as our best strategy. With this, we enter our second phase in which we optimize the network belonging to the best strategy. While the selection just carries out one training and validation run for each strategy, the subsequent optimization step iteratively improves the network belonging to the best sampling strategy via continuous training.

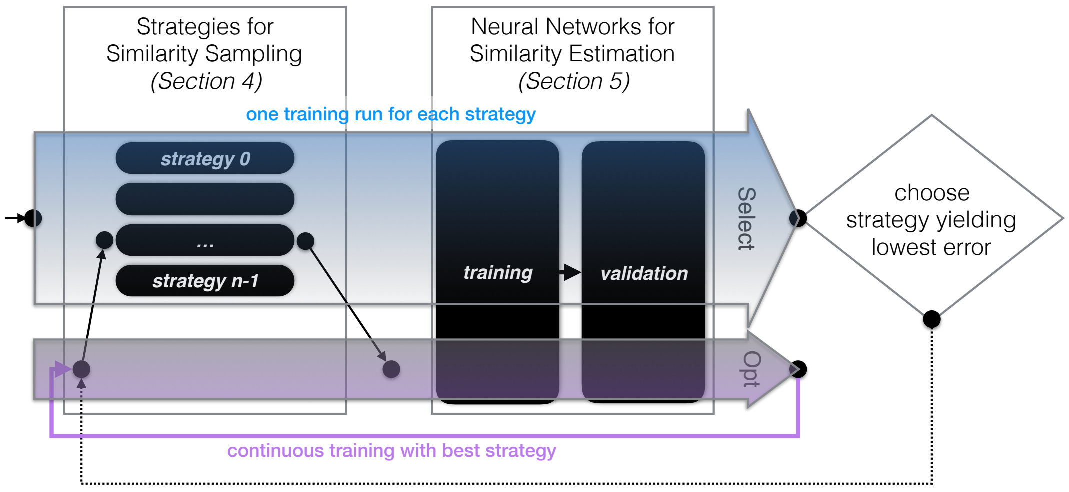

Strategies for Similarity Sampling (Section 4)
Neural Networks for Similarity Estimation (Section 5)
one training run for each strategy
strategy 0
...
strategy n-1
training
validation
Select
choose strategy yielding lowest error
Opt
continuous training with best strategy

Figure 3. Overview of our approach for the adaptive sampling and estimation of similarities with neural networks (cf. Algorithm 2 for a more detailed description). The selection phase (**Select**) chooses the best strategy by carrying out one training and validation run for each strategy. Afterwards, the optimization phase (**Opt**) iteratively improves the network belonging to the best sampling strategy via continuous training with repeatedly updated training data.

4. Strategies for Similarity Sampling

As previously discussed in Section 3.1, we utilize and evaluate progressive approaches to compute similarity information between individual time steps $t \in T$ of a time series T. Here, progressive means that we iteratively add new similarity information (i.e., distances between pairs of time steps). The strategy basically for which time step pair its similarity is computed next. The strategies for determining the next time step pair to compute—subsequently denoted as similarity sampling—considered in this paper are now discussed in the following. Depending on their procedure, they belong to two time types of categories: **similarity pair-based** and **time step-based**.

Similarity pair-based denotes the concept that pairs of time steps can be selected in an arbitrary fashion, and therefore also completely independently from the samples taken so far (naturally, a pair of time steps only needs to be computed once). More formally, this means that the next time step pair (t_0, t_1) to compute the similarity for using metric $d(t_0, t_1)$ is chosen arbitrarily from the full set of time steps T (i.e., $t_0 \in T$ and $t_1 \in T$). Here, the only restriction is that we limit ourselves to $t_0 < t_1$ due the symmetry property of d (cf. discussion in Section 3.1).

In contrast, **time step-based** means that not individual pairs but time steps are progressively added into consideration, and the similarity between all pairwise combinations of considered time steps is computed before selecting new time steps. This means that only a subset $T^* \subset T$ is currently considered. Before adding a new time step $t \in T, t \notin T^*$, we first compute all combinations $t_0 \in T^* \times t_1 \in T^*$ (with $t_0 < t_1$). Only when all pairwise combinations are computed, we add a new time step to T^*.

The different sampling strategies (creating a set of similarity pairs P) we employ and evaluate in this paper on the basis of these different approaches outlined above are as follows ((**1**) and (**2**) follow similarity pair-based, (**3**)–(**7**) follow time step-based).

(1) uniform pair. In this strategy, the goal is to distribute samples in the temporal space $T \times T$ as evenly as possible (in a progressive fashion). Doing this, we start out with a random sample pair $P = \{(t_0, t_1)\}$ ($t_0 \in T$ and $t_1 \in T$). Subsequently, we then iteratively compute the new similarity pair (t_0, t_1) that has the maximum distance to any of the pairs $(t_0, t_1) \in P$ (i.e., to any of the pairs that have been computed so far).

(2) random pair. A time step pair—that has not been computed yet ($\notin P$)—is chosen randomly and processed next.

(3) random time. A random time step is selected that has not yet been considered ($\notin T^*$). As described above, before adding a new time step, first all pairwise combinations of time steps $\in T^*$ are computed before proceeding further.

(4) uniform time. Select the time step in between the largest interval range $t_{i+1} - t_i$ in T^* (with t_i and t_{i+1} denoting subsequent time steps $\in T$). In case there are multiple intervals with the same size, we choose one randomly.

(5) distance-weighted time. Choose a time step randomly (similar to (3)), but the probability of selecting an interval is weighted by $t_{i+1} - t_i - 1$ (akin to the selection criterion in (4)).

(6) similarity time. Consider the distance between two subsequent time steps in T: $d(t_i, t_{i+1})$. Add a time step in the interval with the largest distance.

(7) similarity-weighted time. Select an interval randomly to add a new time step to T^*, with the probability being weighted by $d(t_i, t_{i+1})$.

The different sampling patterns arising from these seven different strategies are exemplified in Figure 4 (distances which have not been computed yet are indicated in red). We achieve a large variety of strategies, following completely random, uniform, and similarity-adaptive approaches.

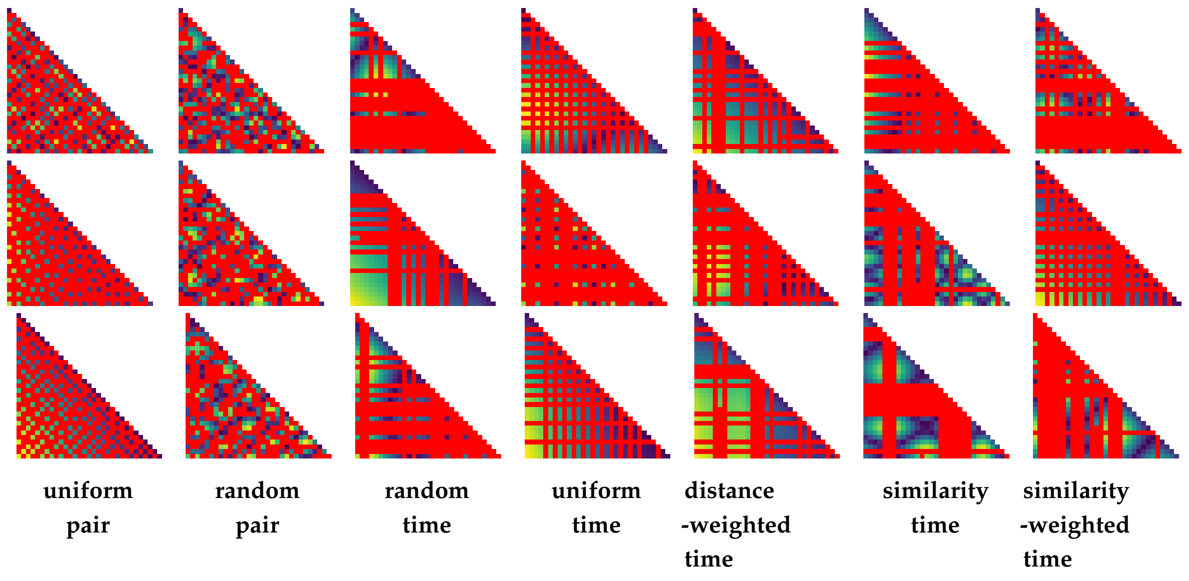


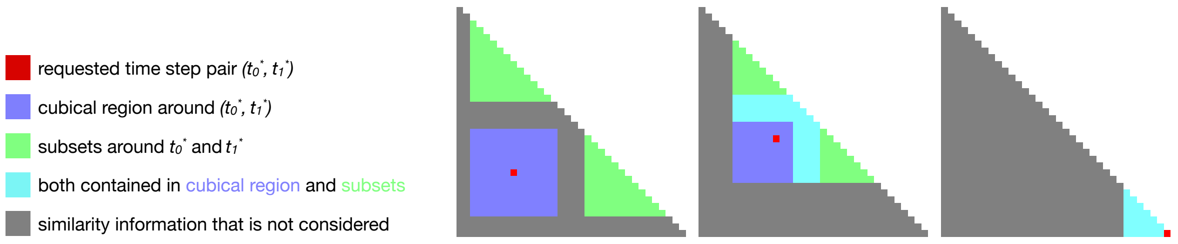
Figure 4. Different sampling strategies by example. Horizontally, we provide the results of a certain sampling strategy, while vertically these strategies are demonstrate by means of different input data sets. Similarity pairs that have not yet been computed by the sampling strategy are indicated in red.

5. Neural Networks for Similarity Estimation

In this section, we first discuss the basic model setup of our neural network (Section 5.1). We then outline how we obtain and generate the data used for its training and validation (Section 5.2). On this basis, we finally discuss our approach to train neural networks and to select the most appropriate respective sampling strategy (Section 5.3).

5.1. Model Setup

Our model is designed to estimate one missing similarity pair (t_0, t_1) on the basis of other available similarity information. The design choices described below are based on empirical tests, informally evaluating different model designs and neural network setups against each other. Note that while our resulting design is the best we found in our tests, we cannot consider it optimal due to the large search space (there is large variety of different ways to configure a neural network alone).



requested time step pair (t0*, t1*)
cubical region around (t0*, t1*)
subsets around t0* and t1*
both contained in cubical region and subsets
similarity information that is not considered

(a) $t_0^* = 8, t_1^* = 25$ **(b)** $t_0^* = 11, t_1^* = 20$ **(c)** $t_0^* = 33, t_1^* = 34$

Figure 5. Illustration of the different time step pairs that are considered for training at the example of different time step pairs (t_0^*, t_1^*) of interest (cf. Section 5.1, **Considered Input Data**).

Considered Input Data. For estimating the similarity information, we utilize a subset of the available similarity information from different regions. In more detail, we jointly consider two types of information in a (temporal) region of extent δ around the requested element (t_0^*, t_1^*) (individually, for each component of the pair, this results in a region of $T_0 = \{t_0 - \delta, \dots, t_0 + \delta\}$ and $T_1 = \{t_1 - \delta, \dots, t_1 + \delta\}$, respectively). An illustration of the considered time steps at different examples is shown in Figure 5.

1. The cubical region around (t_0^*, t_1^*), except for (t_0^*, t_1^*) itself (blue in Figure 5):
$$T_\delta(t_0^*, t_1^*) = T_0 \times T_1 \setminus (t_0^*, t_1^*). \tag{2}$$
This results in $|T_0||T_1| - 1$ elements. This gives the similarity of close pairs in temporal space.
2. Two subsets of the similarity matrix, one around t_0 and one around t_1 (i.e., containing all pairwise combinations of time steps T_0 and T_1, respectively) (green in Figure 5). This results in a total of $|d(T_0)| + |d(T_1)|$ (according to Equation (1)), and gives the similarity to close time steps for each component of the pair.

In total, this accordingly results in I input elements (cf. Figure 5):

$$|I_\delta| = \underbrace{(2\delta + 1)^2 - 1}_{\text{cubical region (1.)}} + \underbrace{2|d(T_\delta)|}_{\text{similarity context per component (2.,based on Equation(1))}} . \tag{3}$$

In cases where no similarity information is available (because it has not been sampled yet or it is out of the temporal range), the respective element gets a dedicated (missing) value of m (in our implementation we set $m = -1$, which clearly indicates a special value as similarity information is generally quantified by positive values). Similarity pair information from the two different types of information may also overlap (e.g., Figure 5b,c), in which case the respective similarity information is considered redundantly as input for the neural network.

Neural Network Structure. Neural networks have three types of layers: input, hidden, and output. There is exactly one input layer, with the number of neurons being determined by the size of the input data. Based on the previous discussion regarding considered input data, this means that we have a total number of $|I_\delta|$ input neurons (Equation (3)). The output layer also consists of exactly one layer. However, here, we just use single neuron as only one specific similarity pair is predicted at a time by our neural network. Regarding the hidden layers, there is much larger degree of freedom : how many hidden layers to actually have in the neural network and how many neurons (and which type) will be in each of these layers. Typically, these decisions have a significant impact on the results that can be achieved, but respective decisions come down to experience and trial-and-error to a certain extent. In this work, we use one hidden layer, with the number neurons equalling the number of neurons in the input layer. According to our experiments, this provides a good trade-off between underfitting and overfitting. For each neuron, we use the sigmoid as an activation function. Note that this is the design that worked best according to our experiments, but we do not consider it to be optimal or any kind of definite solution to the problem (but rather a step toward it).

5.2. Training and Validation Data Preparation

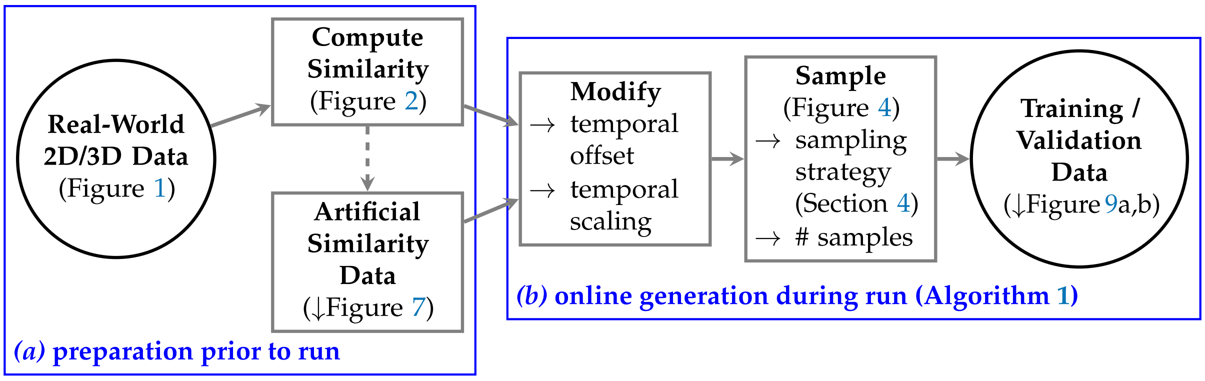

Real-World 2D/3D Data (Figure 1)
Compute Similarity (Figure 2)
Artificial Similarity Data (↓Figure 7)
(a) preparation prior to run
Modify → temporal offset → temporal scaling
Sample (Figure 4) → sampling strategy (Section 4) → # samples
Training / Validation Data (↓Figure 9a,b)
(b) online generation during run (Algorithm 1)

Figure 6. Pipeline for preparing training and validation data.

Algorithm 1 Generation of similarity data sets (for training, validation, and testing)

```
1:  ▷ in: pool of similarity information $P$ (composed of data sets $P_i$, featuring $P_i^{\#\text{ time steps}}$ time steps)
2:  ▷ in: number of sets $n_X$
3:  ▷ in: number of time steps $n_T$
4:  ▷ in: sampling strategy $\lambda$
5:  ▷ out: sets of data $X$
6:  function DATA_SET_GENERATION($P, \lambda, n_X, n_T$)
7:      $X \leftarrow \varnothing$
8:      while $|X| < n_X$ do
9:          ▷ choose data set
10:         $i \leftarrow$ random_ randint($0 \ldots |P| - 1$)
11:         ▷ choose random step size $s$ (i.e., scaling of the time series)
12:         $s \leftarrow$ random_ randint($1 \ldots P_i^{\#\text{ time steps}} / n_T$)
13:         ▷ choose random offset $o$
14:         $o \leftarrow$ random_ randint($0 \ldots \cdot s \cdot P_i^{\#\text{ time steps}} - n_T - 1$)
15:         ▷ generate variant of $P_i$
16:         $P_i^* \leftarrow \varnothing$
17:         for all $e \in [0 \ldots n_T)$ do
18:             $P_i^* \leftarrow P_i^* \cup P_i[o + se]$
19:         end for
20:         ▷ determine number of samples to take, maximum number computed via Equation (1)
21:         $n_S \leftarrow$ random_ randint($0 \ldots |d(n_T)|$)
22:         $X \leftarrow X \cup \{P_i^*, n_S\}$
23:     end while
24:     return $X$
25: end function
```

A large number of adequate training and validation data is crucial for the success when training neural networks. However, the computation of real-world distances is very expensive and can only be done for a few data sets. To overcome this issue, we generate additional artificial data and further modify the similarity information. In detail, we use the following multi-stage approach to generate a large variety of training and validation data (cf. Figure 6).

Real-World 2D/3D Data. As input, we use a set of typical real-world 2D/3D + time data sets (Figure 1).

Compute Similarity. To compute the pairwise distances between different time steps within each series, we use the approach proposed by Frey and Ertl [24,25]. It is used to make it computationally feasible to directly compute the similarity between high-resolution field data sets. Conceptually, it starts with an initial random assignment of so-called source elements from one data set to so-called target elements of the other data set (each element refers to a (scalar) mass unit given at a certain cell/position in the data). Then, this assignment is improved iteratively in the following. In each iteration, source elements exchange respectively assigned target elements under the condition that this improves the assignment. For this, the quality of an assignment is quantified by d, that essentially computes the sum of weighted distances of the assignments. Here, assignments are weighted by the scalar quantity that is transported. We use this value d directly (on the basis of Euclidean distances) to quantify the distance between the respective time steps. The respective results are shown in Figure 2. Please refer to Frey and Ertl [24,25] for a more detailed discussion.

Artificial Similarity Data. Only using a small number of data sets is not sufficient to cover the large variety of typical patterns of similarity information in general, and might also be dangerous in terms of training the network regarding the concrete data rather than generalizing for similarity estimation. Therefore, we added further, synthetically-generated time series data to supplement this. Here, the idea is to mimic the typical patterns that we have seen occurring in the similarity data, yet providing a larger variety to yield better generalization characteristics after learning. For this, we used the following equation ψ for $\bar{t} \in [0, 1)$, and three random values $\rho_0, \rho_1, \rho_2 \in [0, 1)$:

$$\psi(\bar{t}) = (\bar{t}\rho_0 + 0.5(1-\bar{t}))\sin(\bar{t}\rho_1 + \rho_2(1-\bar{t})) \tag{4}$$

We then compute similarity information from these, and use it during training and validation (Figure 7).

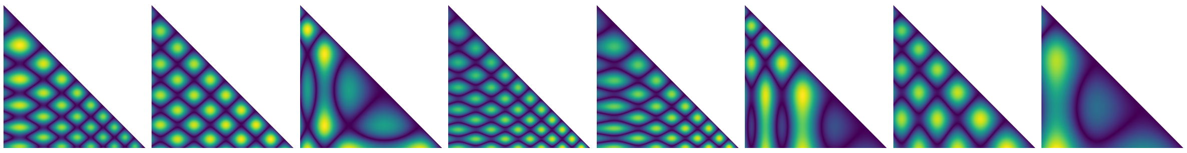


Figure 7. Input similarity information from synthetic data (Equation (4)).

Modify. We do not use the obtained similarity information directly, but randomly offset and scale the time series to get numerous variations on the basis of the available data. We outline our approach to prepare the training data X by means of Algorithm 1 (validation data is generated accordingly). We randomly pick data sets from our collection of real-world and artificial data (Line 10). To modify the data, we randomly choose a scaling factor s, that basically defines the step size with which time steps are considered (Line 12). Then, we use a random offset, which basically determines the first time step that is considered in a time series (Line 14). Finally, we employ this information to generate a new training element P_i^* (Lines 16–19), each one consisting of n_T time steps (we use $n_T = 35$ throughout this work).

Sample. We then take a random number of samples s from the modified similarity data using the respective sampling strategy (cf. discussion in Figure 4, Line 21).

Training / Validation Data. Finally, this yields the data that can be used for training and validation of the neural network. In more detail, each training / validation data element consists of a pair: (1) the original similarity data after **Modify**, and (2) the respective data after **Sample**. Each (1) and (2) contains pairwise similarity information between n_T time steps (a portion of this information has been removed from (2)).

5.3. Similarity Estimation

Algorithm 2 Our approach to estimate missing similarity information based on neural networks (see Figure 3 for a conceptual overview).

```
1:  ▷ in: pool of similarity information $P$
2:  ▷ in: number of sets $n_X$
3:  ▷ in: number of time steps $n_t$
4:  ▷ out: model for similarity estimation $\Theta$
5:  function TRAIN_SIMILARITY_ESTIMATION($P$)
6:      $Y_{\text{train}} \leftarrow$ DATA_SET_GENERATION($P, n_{\text{train}}, n_T$)
7:      $Y_{\text{validate}} \leftarrow$ DATA_SET_GENERATION($P, n_{\text{validate}}, n_T$)
8:      ▷ Select Sampling Model $\Lambda$
9:      $\check{c} \leftarrow \infty$
10:     ▷ Loop over all sampling strategies $\Lambda$ (as discussed in Section 4)
11:     for all $\lambda \in \Lambda$ do
12:         ▷ obtain $n_\lambda$ samples using sampling strategy $\lambda$
13:         $X^{\lambda}_{\text{train}} \leftarrow \lambda(Y_{\text{train}})$
14:         ▷ we use the Adam optimizer [40] for training the neural network
15:         $\Theta_\lambda \leftarrow$ TRAIN($X^{\lambda}_{\text{train}}, Y_{\text{train}}$)
16:         ▷ do validation
17:         $X^{\lambda}_{\text{validate}} \leftarrow \lambda(Y_{\text{validate}})$
18:         $c \leftarrow$ evaluate($X_{\text{validate}}, Y_{\text{validate}}$)
19:         if $c < \check{c}$ then
20:             $\check{c} \leftarrow c; \check{\lambda} \leftarrow \lambda$
21:         end if
22:     end for
23:     ▷ continue training with best selected model
24:     loop
25:         $Y_{\text{train}} \leftarrow$ DATA_SET_GENERATION($P, n_{\text{train}}, n_T$)
26:         $X^{\check{\lambda}}_{\text{validate}} \leftarrow \check{\lambda}(Y_{\text{validate}})$
27:         $\Theta_{\check{\lambda}} \leftarrow$ TRAIN($X^{\check{\lambda}}_{\text{train}}, Y_{\text{train}}$)
28:     end loop
29:     return $\Theta$
30: end function
```

Our overall approach to use neural networks to estimate missing similarity information and to select sampling strategies has been conceptually outlined already in Figure 3. In the following, we now aim to describe it in more detail by means of Algorithm 2.

First of all, we generate separate sets for training and validation using the procedure described above (Lines 6 and 7). On this basis, we then aim to determine the sampling strategy with the lowest cost $\check{c}$ (Lines 9–22). For each sampling strategy $\lambda \in \Lambda$ (Line 11), we then obtain a sampled (i.e., sparse) variant of the similarity information (Line 13). We then use this as input for training (Line 15). During validation (Lines 17–18), we determine a cost c that we then compare against the cost obtained by other strategies. If it is smaller than the smallest cost $\check{c}$ determined so far (Line 19), we save the respective model λ of the respective strategy as the best one so far (Line 20). After testing all models, we continue refining the model that corresponds to the sampling strategy that led to smallest validation cost $\check{c}$ (Line 27).

6. Results

In this section, we first discuss our evaluation setup (Section 6.1). We then evaluate the results with different sampling strategies (Section 6.2) as well as similarity estimation with neural networks for the selected strategy (Section 6.3). Finally, we discuss properties and limitations of our approach (Section 6.4).

6.1. Evaluation Setup

Parameters, Software and Hardware Setup. For our implementation, we use Python and TensorFlow [41] (r0.11). For training, we used the GPU implementation on the basis of CUDA using a GTX1070 on an Ubuntu 16.04 system with 32GB of RAM and an Intel Core i7-4770 CPU. Furthermore, we employ a batch size of 4096 and 1024 training iterations. In total, this means that there are four million training cases in each epoch. For validation, we use 16384 cases. As discussed above, each individual case consists of a reference (i.e., a modified version of a real-world or a synthetic data set) as well as a sampled version of it. In our evaluation, we randomly vary the number of samples such that they cover between 10% and 50% of all pairwise similarity information. We use squared distances to assess the difference of the estimated similarity to the reference. In this work, as mentioned above, we evaluate our approach by considering time windows of size $n_T = 35$. In total, the generation of training sets, training, and validation takes around six hours for testing each sampling strategy using our setup described above. Note that overall the goal here is to train a network that is able to predict similarity for a wide range of temporal data, which is why we train and validate considering a large variety of generated training data. This means that the training process only needs to be done once, and the resulting neural network can then be applied to estimating similarity data as-is. Evaluating the trained network can be done very quickly and yields comparable performance to other types of estimation considering a similar amount of data for estimation (e.g., via inverse distance weighting as discussed below).

Comparison against Inverse Distance Weighting for Similarity Estimation. We compare our approach for estimating missing similarities with neural networks against a standard approach for scattered data interpolation, namely inverse distance weighting. Here, the similarity information corresponding to time step pairs $d'(t_0, t_1)$ is calculated with a weighted average of the values available for the known pairs. The neighborhood considered here is the same as is used by the neural network (i.e., as specified in Equation (2)). With this, the value estimation $d'(t_0, t_1)$ for a missing (i.e., not yet sampled) value is computed as follows (D denotes a map of previously computed similarity information, with m being returned for unknown pairs):

$$d'_\delta(t_0, t_1) = \frac{\sum_{(\bar{t_0}, \bar{t_1}) \in T_\delta(t_0, t_1)} \omega(\bar{t_0}, \bar{t_1}) D(\bar{t_0}, \bar{t_1})}{\sum_{(\bar{t_0}, \bar{t_1}) \in T_\delta(t_0, t_1)} \omega(\bar{t_0}, \bar{t_1})}, \tag{5}$$

with

$$\omega(t_0, t_1) = \begin{cases} \left(\frac{1}{\sqrt{(t_0 - t_0^*)^2 + (t_1 - t_1^*)^2}} \right)^p & \text{, if } D(t_0, t_1) \neq m \\ 0 & \text{, else.} \end{cases} \tag{6}$$

Here, p is a positive real number that specifies the power parameter. Weight decreases as distance increases from the interpolated points. Larger values for p assign greater influence to values closest to the interpolated point, with the result converging toward nearest neighbor interpolation for large values of p.

6.2. Sampling Strategies

For evaluating the different sampling strategies (cf. Section 4) regarding their performance in the context of a neural networks for similarity estimation, we train a neural network over 2048 epochs with respectively sampled data.

For each sampling strategy, we further compare the validation results of the neural network to the results achieved with inverse distance weighting for different power parameters. The respective results are shown in Figure 8. Most prominently, it can be seen that the sampling strategies relatively perform similarly across all estimation approaches: **uniform pair** and **uniform time** yield the best results (lowest validation cost), while **random time** and **similarity time** yield the worst results here. On the basis of the different sampling patterns in Figure 4, we assume that the main reason behind this are the resulting larger temporal regions in which no similarity information is available. For these, it is much more difficult across all similarity estimation approaches to yield reasonably good results. Note that we consider a number of samples that is randomly chosen to be between 10% and 50% of the full sampling. In preliminary tests with a larger number of samples, adaptive approaches performed much better relatively. This indicates that a more complex combination of strategies (or more advanced sampling strategies overall) could be worthwhile to consider in this context. However, a closer investigation and evaluation of this remains for future work.

The best result overall in our evaluation setting is achieved by our neural network-based similarity estimation with the **uniform time** sampling strategy. Not only in this case but within each sampling strategy, it can be seen that our neural network-based approach consistently yields better results (i.e., lower validation cost) than any inverse distance weighting variant. Please refer to the upcoming section for a closer discussion of the different reasons behind this at the example of the **uniform time** sampling strategy.

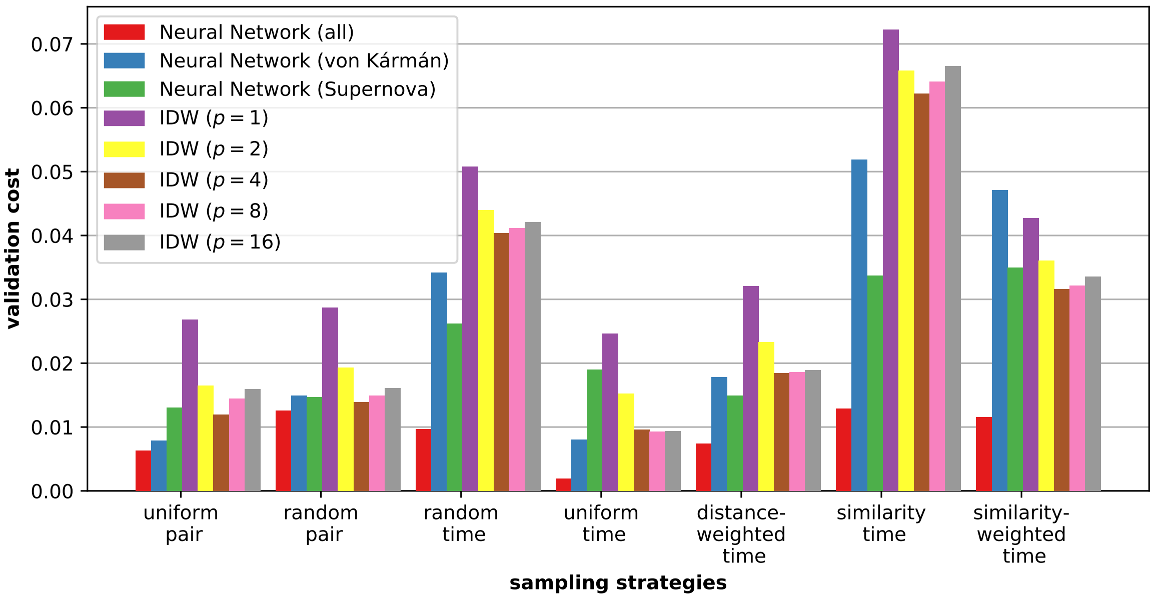


Figure 8. Results of different adaptive sampling strategies and interpolation schemes after the initial phase of sampling strategy selection (**Select** in Figure 3). Respective costs are given for the estimation with the trained neural network as well as inverse distance interpolation (*IDW*) for different power parameters p. For the neural network, not only the results for training with all data sets (*Neural Network (all)*) are presented, but also the validation costs are provided for neural networks that have only be trained with either the von Kármán data set (*Neural Network (von Kármán)* on the basis of Figure 2d) or the Supernova data set (*Neural Network (Supernova)* on the basis of Figure 2e).

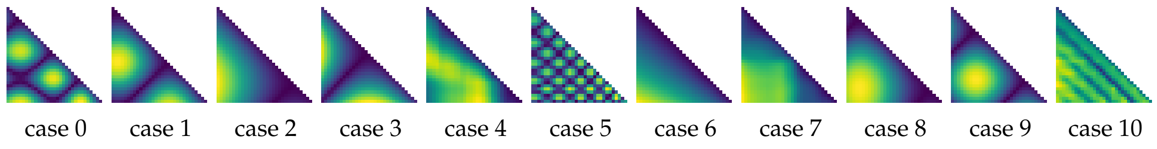
For analyzing the utility of using a variety of different data sets for training, we also include the results of networks that have just been trained on the basis of one data set. For this evaluation, we use the von Kármán data set (*Neural Network (von Kármán)* in Figure 8, employing similarity information from Figure 2d) as well as the Supernova data set (*Neural Network (Supernova)*, on the basis of Figure 2e). In both cases, still the same total number of training data sets is generated via modification and sampling. This means that the only difference is that just a single real-world data set (and no artificial similarity data) is employed for training, but the same validation process is used as for *Neural Network (all)* (i.e., all data sets are always considered for validation). It can be seen in Figure 8 that the respective neural networks trained with a single data set deliver worse results than the neural network that has been trained more diversely with all data sets (*Neural Network (all)*). However, they still produce reasonable results that are comparable to the quality generated by inverse distance weighting. Comparing *Neural Network (von Kármán)* and *Neural Network (Supernova)* against each other, it can be seen that their relative performance depends significantly on the sampling strategy as well. Essentially, this indicates that how successful different sampling strategies are also depends on the type of properties and characteristics of the similarity information that is employed for training. Among others, this supports the approach—as discussed in Section 3.3—of taking a variety of strategies into account and using an automatic approach to select the best one for a provided collection of data sets of interest. A more exhaustive evaluation of respective properties remains for future work.

6.3. Similarity Estimation

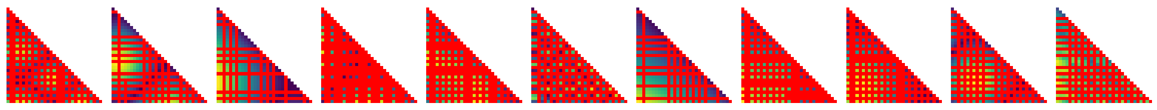
Next, we discuss the results of similarity estimation with the uniform time sampling strategy that has been determined to deliver the best results in our setup. Reference similarity information, samples, and the reconstructed information with the estimated similarities for our neural network as well as for inverse distance interpolation are shown in Figure 9 (at the example of a subset of the validation set). Overall, as reflected by the small cost/error value, it can be seen that even in cases where a large portion of the data is missing, the trained network performs well in filling in the missing information.

Good results can be achieved with our neural network-based approach over a large variety of cases. Despite potentially only a fraction of the similarity information being available and/or temporal changes occurring at a high rate, we are still able to yield a good approximation of the actual values.

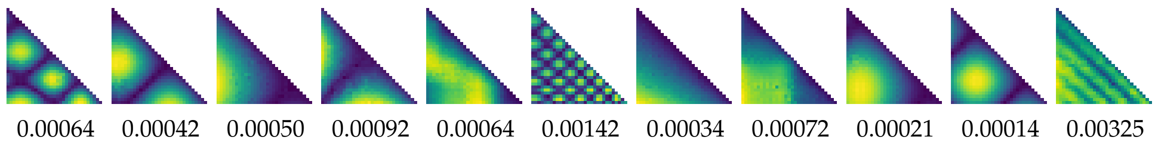
In comparison, inverse distance weighting struggles particularly in the case of a higher rate of changes (e.g., case 5 and case 10). As we consider a fairly large neighborhood ($\delta = 6$), the lower power variants for inverse distance interpolation that also give further away samples a significant weight yield insufficient results, in particular for the cases with a large variation (i.e., $p = 1$ and $p = 2$). In turn, a large power parameter ($p = 16$) effectively only considers the closest points, which yields blocky (non-smooth) results which is particularly noticeable in some smoother cases (e.g., cases 7 and 8). The best performance of inverse distance weighting overall is achieved in-between with $p = 4$ and $p = 8$, that shares both issues of a high and a low power parameter, yet to a lesser extent. However, generally inferior results are achieved in comparison to the similarity estimation by the neural network.



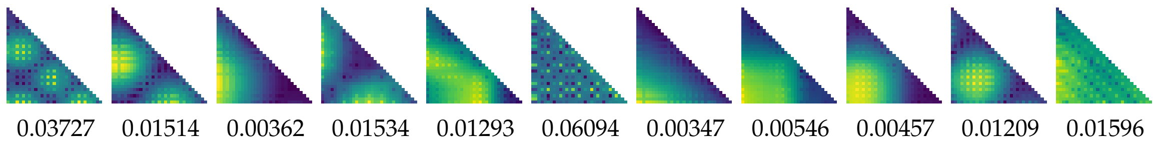
(**a**) Reference.



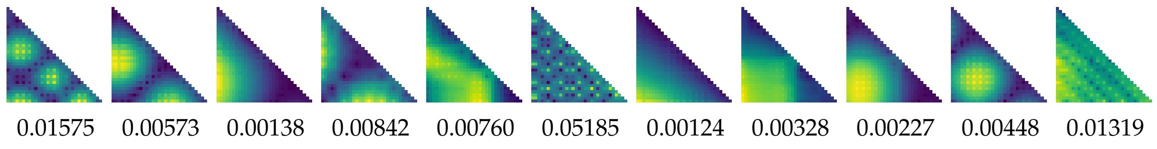
(**b**) Sampling (missing colors indicated in red).



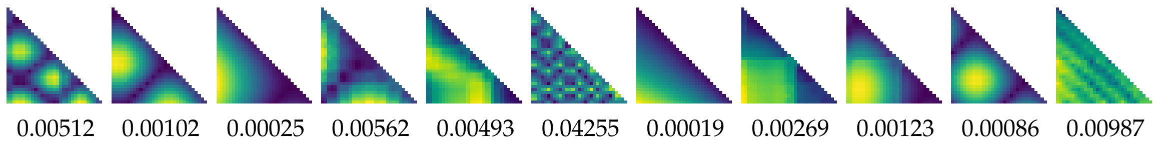
(**c**) Neural Network (all).



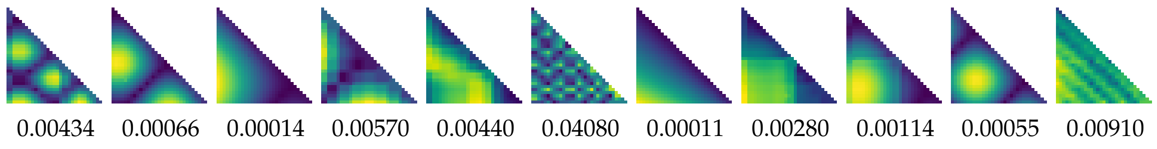
(**d**) Inverse Distance Weighting ($p = 1$).



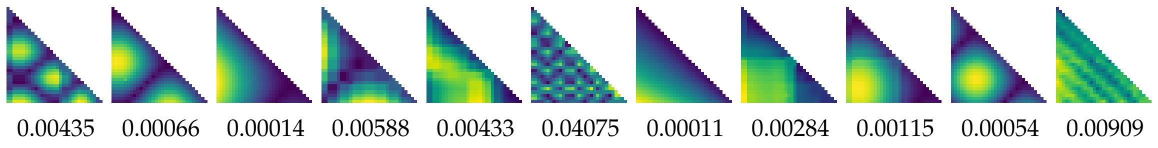
(**e**) Inverse Distance Weighting ($p = 2$).



(**f**) Inverse Distance Weighting ($p = 4$).



(**g**) Inverse Distance Weighting ($p = 8$).



(**h**) Inverse Distance Weighting ($p = 16$).

Figure 9. Reference, sampling, and estimation of similarities for different interpolation strategies. The numbers below the plots with estimated similarity give the respective validation cost ((**c**)–(**h**)).

6.4. Discussion

While the approach presented in this paper delivers promising results, there are also a variety of limitations and shortcomings that we aim to briefly discuss in the following. On this basis, we also indicate plans and directions for future work to overcome some of these limitations.

First of all, there is a large parameter space in setting up, configuring, training, etc. of neural networks. Typically, these decisions have a significant impact on the results that can be achieved, but respective decisions come down to experience and trial-and-error to a certain extent. The paper at hand presents the best design of a network for spatio-temporal similarity estimation we determined so far according to our experiments. However, we do not consider it to be optimal or any kind of definite solution to the problem. To find a better solution, we aim to systematically try different types of neurons, different numbers of layers and numbers of neurons in each layer, different batch sizes and learning rates, etc. As discussed above, we also found that more advanced sampling similarity strategies beyond the rather basic ones considered in this paper could be worthwhile to pursue and potentially yield better results with the same number of samples. Likewise, although the formulation of the synthetic data is comparably simple (Equation (4)), it can create a variety of different structures we see in practice (in combination with the additional modifications applied to the data). However, a more elaborate method could be able to more comprehensively represent a wider variety of time series characteristics. To develop such a more advanced artificial representation of similarity data in future work, we aim to systematically consider a large and diverse set of similarity information in different kinds of time series data.

Conceptually, the task that we solve with our neural network can be seen to be the generation of the full set of pairwise similarity information from a given set of samples. This problem can be described as a general reconstruction or interpolation problem from sparse samples, and essentially a wide variety of methods can be applied to that problem. Naturally, this includes inverse distance weighting that we use for the sake of comparison above, but also more complex techniques from signal processing like compressed sensing. However, note that there are special properties that differentiate time series similarity data from standard signals or images. These properties depend on the specific metric that is used for assessing the difference between two time steps. For instance, for the distance operator used in this paper, there is the symmetry property (cf. Section 3.1), and $d(t_a, t_b) + d(t_b, t_c) \approx d(t_a, t_c)$ holds for linear behavior in between t_a and t_c. We consider our basic approach described in this paper to be generally applicable for the training and estimation of different types of data and metrics, as we do not explicitly encode these properties, but rather aim to let the network adapt to this through training. The only exception to this is the symmetry property that we explicitly exploit for the sake of efficiency (resulting in the triangle-shaped similarity plots), but the extension to non-symmetric metrics is straight-forward. In general, we assume that the more cross-relations there are in the data, the more do we benefit from using a learning-based approach that is able to capture these respective properties. However, a closer evaluation of these aspects is subject to future work. Also, while in this paper we limit ourselves to comparing our neural network-based approach to inverse distance weighting methods (that are the most popular approach in the field of computer graphics and visualization for scattered data interpolation), advanced approaches from different domains (particularly signal processing) could also yield good results in our considered scenario. A thorough comparison against (different implementations of) these methods also remains for future work.

Furthermore, we aim to evaluate the utility of our approach in real-world visualization scenarios. As discussed above, there is a variety of visual analysis applications that can directly use the estimated similarity information for automatic selection and aggregation of data, and we would like to evaluate the impact of the quality of the distance estimation on the final visualization result. While we believe that our approach is generally applicable in terms of methodology, we particularly aim to more thoroughly evaluate the generality of a trained network. In our evaluation (cf. Figure 8 and respective discussion in Section 6.2), we already consider the extreme example of a network that has just been trained on the basis of one data set (that has however been modified as discussed in Section 5.2). Here, it can be seen that, while delivering significantly worse results than our comprehensively trained network, it still yields decent results that are comparable in quality to the reference inverse distance weighting techniques. While this can be interpreted as a small indicator that we are able to achieve good results for general similarity information from a small set of data sets, a more extensive evaluation is required to thoroughly analyze respective properties.

7. Conclusions

In this work, we presented different strategies for the progressive computation of similarity information in spatio-temporal data, along with an approach for estimating missing distance information. For similarity estimation, we proposed to use a neural network design that directly takes the already available similarity information of a time series into account. We then automatically determined the sampling strategy that yields the best result in combination with respectively trained networks for estimation. For training and validation, we used a variety of time-dependent 2D and 3D data from simulations and measurements as well as artificially generated data.

We could demonstrate that we achieve good results already with our proposed approach, with further improvements being subject to future work. In particular, we further aim to further explore the huge parameter space inherent to the setup and training of neural networks by systematically testing different types of neurons, different numbers of layers and numbers of neurons in each layer, different batch sizes and learning rates, etc. to further improve our results. We also plan to to evaluate the impact of the quality of estimated distance estimation on the final result of different types of visualization applications. Finally, we aim to compare our approach to a larger variety of alternative approaches for similarity estimation, develop more advanced techniques for artificial data generation, and conduct a more comprehensive evaluation regarding generalization properties.

Acknowledgments: The author would like to thank the German Research Foundation (DFG) for supporting the project within project A02 of SFB/Transregio 161 and the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Joshi, A.; Rheingans, P. Illustration-inspired techniques for visualizing time-varying data. In Proceedings of the VIS 05, IEEE Visualization, Minneapolis, MN, USA, 23–28 October 2005; pp. 679–686.
2. Lee, T.Y.; Shen, H.W. Visualization and Exploration of Temporal Trend Relationships in Multivariate Time-Varying Data. *IEEE Vis. Comput. Graph.* **2009**, *15*, 1359–1366.
3. Joshi, A.; Rheingans, P. Evaluation of illustration-inspired techniques for time-varying data visualization. *Comput. Graph. Forum* **2008**, *27*, 999–1006.
4. Woodring, J.; Wang, C.; Shen, H.W. High dimensional direct rendering of time-varying volumetric data. In Proceedings of the IEEE Visualization, Seattle, WA, USA, 19–24 October 2003; pp. 417–424.
5. Balabanian, J.P.; Viola, I.; Möller, T.; Gröller, E. Temporal Styles for Time-Varying Volume Data. In Proceedings of the 3DPVT'08—The Fourth International Symposium on 3D Data Processing, Visualization and Transmission, Atlanta, GA, USA, 18–20 June 2008; Gumhold, S., Kosecka, J., Staadt, O., Eds.; 2008; pp. 81–89.
6. Bach, B.; Dragicevic, P.; Archambault, D.; Hurter, C.; Carpendale, S. A Descriptive Framework for Temporal Data Visualizations Based on Generalized Space-Time Cubes. *Comput. Graph. Forum* **2016**, doi:10.1111/cgf.12804.
7. Fang, Z.; Möller, T.; Hamarneh, G.; Celler, A. Visualization and Exploration of Time-varying Medical Image Data Sets. In Proceedings of the Graphics Interface 2007, Montreal, QC, Canada, 28–30 May 2007; ACM: New York, NY, USA, 2007; pp. 281–288.
8. Wang, C.; Yu, H.; Ma, K.L. Importance-Driven Time-Varying Data Visualization. *IEEE Vis. Comput. Graph.* **2008**, *14*, 1547–1554.

9. Widanagamaachchi, W.; Christensen, C.; Bremer, P.T.; Pascucci, V. Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In Proceedings of the IEEE Symposium on Large Data Analysis and Visualization (LDAV), Seattle, WA, USA, 14–15 October 2012; pp. 9–17.
10. Lee, T.Y.; Shen, H.W. Visualizing time-varying features with TAC-based distance fields. In Proceedings of the 2009 IEEE Pacific Visualization Symposium, Beijing, China, 20–23 April 2009; pp. 1–8.
11. Silver, D.; Wang, X. Tracking and Visualizing Turbulent 3D Features. *IEEE Vis. Comput. Graph.* **1997**, *3*, 129–141.
12. Ji, G.; Shen, H.W. Feature Tracking Using Earth Mover's Distance and Global Optimization. *Pac. Graph.* **2006**.
13. Weber, G.; Dillard, S.; Carr, H.; Pascucci, V.; Hamann, B. Topology-Controlled Volume Rendering. *IEEE Vis. Comput. Graph.* **2007**, *13*, 330–341.
14. Narayanan, V.; Thomas, D.M.; Natarajan, V. Distance between extremum graphs. In Proceedings of the IEEE Pacific Visualization Symposium, Hangzhou, China, 14–17 April 2015; pp. 263–270.
15. Schneider, D.; Wiebel, A.; Carr, H.; Hlawitschka, M.; Scheuermann, G. Interactive Comparison of Scalar Fields Based on Largest Contours with Applications to Flow Visualization. *IEEE Vis. Comput. Graph.* **2008**, *14*, 1475–1482.
16. Rubner, Y.; Tomasi, C.; Guibas, L. The Earth Mover's Distance as a Metric for Image Retrieval. *Int. J. Comput. Vis.* **2000**, *40*, 99–121.
17. Tong, X.; Lee, T.Y.; Shen, H.W. Salient time steps selection from large scale time-varying data sets with dynamic time warping. In Proceedings of the 2012 IEEE Symposium on Large Data Analysis and Visualization (LDAV), Seattle, WA, USA, 14–15 October 2012; pp. 49–56.
18. Seshadrinathan, K.; Bovik, A.C. Motion Tuned Spatio-temporal Quality Assessment of Natural Videos. *IEEE Trans. Image Process.* **2010**, *19*, 335–350.
19. Correa, C.D.; Ma, K.L. Dynamic Video Narratives. *ACM Trans. Graph.* **2010**, *29*, 88.
20. Lu, A.; Shen, H.W. Interactive Storyboard for Overall Time-Varying Data Visualization. In Proceedings of the 2008 IEEE Pacific Visualization Symposium, Kyoto, Japan, 5–7 March 2008; pp. 143–150.
21. Post, F.H.; Vrolijk, B.; Hauser, H.; Laramee, R.S.; Doleisch, H. The State of the Art in Flow Visualisation: Feature Extraction and Tracking. *Comput. Graph. Forum* **2003**, *22*, 775–792.
22. McLoughlin, T.; Laramee, R.S.; Peikert, R.; Post, F.H.; Chen, M. Over Two Decades of Integration-Based, Geometric Flow Visualization. *Comput. Graph. Forum* **2010**, *29*, 1807–1829.
23. Garces, E.; Agarwala, A.; Gutierrez, D.; Hertzmann, A. A Similarity Measure for Illustration Style. *ACM Trans. Graph.* **2014**, *33*, 93.
24. Frey, S.; Ertl, T. Progressive Direct Volume-to-Volume Transformation. *IEEE Trans. Vis. Comput. Graph.* **2017**, *23*, 921–930.
25. Frey, S.; Ertl, T. Fast Flow-based Distance Quantification and Interpolation for High-Resolution Density Distributions. In Proceedings of the EG 2017 (Short Papers), Lyon, France, 24–28 April 2017.
26. Frey, S.; Ertl, T. Flow-Based Temporal Selection for Interactive Volume Visualization. *Comput. Graph. Forum* **2016**, doi:10.1111/cgf.13070.
27. Marwan, N.; Carmenromano, M.; Thiel, M.; Kurths, J. Recurrence plots for the analysis of complex systems. *Phys. Rep.* **2007**, *438*, 237–329.
28. Vasconcelos, D.; Lopes, S.; Viana, R.; Kurths, J. Spatial recurrence plots. *Phys. Rev. E* **2006**, *73*, doi:10.1103/PhysRevE.73.056207.
29. Marwan, N.; Kurths, J.; Saparin, P. Generalised recurrence plot analysis for spatial data. *Phys. Lett. A* **2007**, *360*, 545–551.
30. Bautista-Thompson, E.; Brito-Guevara, R.; Garza-Dominguez, R. RecurrenceVs: A Software Tool for Analysis of Similarity in Recurrence Plots. In Proceedings of the Electronics, Robotics and Automotive Mechanics Conference 2008, Morelos, Mexico, 30 September–3 October 2008; pp. 183–188.
31. Frey, S.; Sadlo, F.; Ertl, T. Visualization of temporal similarity in field data. *IEEE Trans. Vis. Comput. Graph.* **2012**, *18*, 2023–2032.
32. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; The MIT Press: Cambridge, MA, USA, **2016**.
33. Hu, H.; Holman, P.M.; de Haan, G. Image interpolation using classification-based neural networks. In Proceedings of the IEEE International Symposium on Consumer Electronics, Reading, UK, 1–3 September 2004; pp. 133–137.
34. Plaziac, N. Image Interpolation Using Neural Networks. *IEEE Trans. Image Proccess.* **1999**, *8*, 1647–1651.

35. Chen, C.H.; Kuo, C.M.; Yao, T.K.; Hsieh, S.H. Anisotropic Probabilistic Neural Network for Image Interpolation. *J. Math. Imag. Vis.* **2014**, *48*, 488–498.
36. Guillaumin, M.; Verbeek, J.J.; Schmid, C. Is that you? Metric learning approaches for face identification. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision, Kyoto, Japan, 29 September–2 October 2009; pp. 498–505.
37. Kulis, B. Metric Learning: A Survey. *Found. Trends Mach. Learn.* **2013**, *5*, 287–364.
38. Davis, J.V.; Kulis, B.; Jain, P.; Sra, S.; Dhillon, I.S. Information-theoretic metric learning. In Proceedings of the 24th International Conference on Machine Learning, Corvalis, OR, USA, 20–24 June 2007; pp. 209–216.
39. Velten, A.; Wu, D.; Jarabo, A.; Masia, B.; Barsi, C.; Joshi, C.; Lawson, E.; Bawendi, M.; Gutierrez, D.; Raskar, R. Femto-photography: Capturing and Visualizing the Propagation of Light. *ACM Trans. Graph.* **2013**, *32*, 44.
40. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *axiv* **2014**, arXiv:1412.6980.
41. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Available online: tensorflow.org (accessed on 22 August 2017).