DOI: 10.1111/cgf.14548
Eurographics Conference on Visualization (EuroVis) 2022
R. Borgo, G. E. Marai, and T. Schreck
(Guest Editors)

Volume 41 (2022), Number 3

SimilarityNet: A Deep Neural Network for Similarity Analysis Within Spatio-temporal Ensembles

Karim Huesmann and Lars Linsen

University of Münster, Germany

Abstract

Latent feature spaces of deep neural networks are frequently used to effectively capture semantic characteristics of a given dataset. In the context of spatio-temporal ensemble data, the latent space represents a similarity space without the need of an explicit definition of a field similarity measure. Commonly, these networks are trained for specific data within a targeted application. We instead propose a general training strategy in conjunction with a deep neural network architecture, which is readily applicable to any spatio-temporal ensemble data without re-training. The latent-space visualization allows for a comprehensive visual analysis of patterns and temporal evolution within the ensemble. With the use of SimilarityNet*, we are able to perform similarity analyses on large-scale spatio-temporal ensembles in less than a second on commodity consumer hardware. We qualitatively compare our results to visualizations with established field similarity measures to document the interpretability of our latent space visualizations and show that they are feasible for an in-depth basic understanding of the underlying temporal evolution of a given ensemble.*

1. Introduction

Spatio-temporal simulations of natural phenomena based on mathematical models are ubiquituous in the sciences. Typically, multiple simulation runs are performed to investigate the model's dependence on different initial configurations or input parameters. The analysis of such simulation ensembles requires visualization methods that handle the ensemble as a whole. Common analysis tasks such as detecting clusters of simulation runs or outliers, investigating the temporal evolution of simulation runs such as converging or diverging behavior, or the detection of key timesteps with sudden changes can be supported by computing field similarities and mapping the similarity space to a visual space. Different field similarity measures have been proposed in this regard [FL19, NNN11, FML15, EHNP04, STS06] to capture different similarity aspects. A drawback of this approach is that a similarity matrix whose entries contain all pairwise field similarities needs to be computed to define the similarity space. For large ensembles with a high number of timesteps and runs, the matrix becomes big and its computation becomes expensive.

Latent spaces of deep neural networks have shown their effectiveness to capture semantics in data within many application scenarios [SWG*19, HTW18, TFE21]. In this paper, we aim at exploiting these capabilities to visualize temporal evolutions within spatio-temporal simulation ensembles using a latent space visualization. We propose a respective deep neural network architecture that is targeted at creating latent spaces, which capture characteristics of spatio-temporal data and can be mapped to a visual space. Moreover, we propose a novel training strategy that allows us to apply our approach to any spatio-temporal ensemble dataset from any application domain without re-training the neural network with data-specific inputs. Our training strategy is based on the creation of a set of random spatial data that evolve over time with respect to some basic functions. We show that when having trained the network with this dataset, it is capable of capturing respective trends in unseen data.

Our deep neural network, to which we refer as *SimilarityNet*, is based on a combination of a transformer and an autoencoder architecture. Transformer networks are currently the best performing network architecture for sequence-to-sequence tasks like language translation, image captioning, conversational models, and text summarization. Therefore, we use a transformer layer to handle sequential temporal data. This layer allows us to handle long sequences of varying lengths. The latent space is generated using an elementary autoencoder structure.

With the help of SimilarityNet, we are able to create 1D embeddings of unseen spatio-temporal datasets that can be used for similarity analysis of individual time steps. By plotting the 1D embeddings over time, the temporal evolution of a simulation run is depicted in a 2D layout, which effectively supports the comparison of different simulation runs within an ensemble.

Our main contributions can be summarized as follows:

- A deep neural network architecture, whose latent space cap-

tures the similarities in spatio-temporal ensemble data without the need for an explicit definition of a field similarity measure.
- A training strategy that makes SimilarityNet generally applicable even to unseen data.
- A latent-space visualization to show temporal evolutions of simulation runs within an ensemble.
- Experiments on synthetic and multiple real-world ensemble datasets to document the interpretability of the latent-space visualizations.
- A technique to efficiently create 1D embeddings on large datasets.

2. Related Work

2.1. Multi-Run Spatio-temporal Similarity Visualization

Research in the analysis of spatio-temporal multi-run simulations has brought forth many different visualization approaches that address the complexity of this type of data in different ways [WHLS19]. Ensemble data emerge from many scientific domains, where some fields require unique visualization frameworks [FFH21, LHFL19]. A variety of visualizations aim to understand the influence of simulation parameters better [KSHW21, Ham94, BPFG11]. Here, many works focus on visually representing the similarity of different runs [FML15]. This helps the analyst to get an overview of the temporal evolution of individual runs as well as the ensemble as a whole. The identification of groups of high similarity or clusters within an ensemble provides insight into how the variation of simulation parameters affects their temporal evolution. For the generation of similarity plots, different strategies for dimensionality reduction methods can be applied to define embeddings of the similarity space. The generation of the similarity space depends on the choice of a distance measures, where different distance measures can highlight different characteristics of the given ensemble [FJC*19, FL19]. The choice of an appropriate distance measure, thus, may significantly impact the analysis outcome. We propose SimilarityNet, where the generated similarity plots stem from a latent space visualization that is learned from data without choosing and applying a specific distance measure. Moreover, computing distance matrices to be fed to dimensionality reduction methods to create the similarity space embeddings is actually computationally costly. SimilarityNet, on the other hand, allows for a direct mapping of data into the latent space, which is computationally light-weight. The resulting visualizations we obtain from SimilarityNet are 1D embeddings plotted over time. Several works have successfully documented that analyzing the evolution of simulation runs as 1D embeddings over a time axis can help understanding temporal patterns in a given simulation run or an ensemble [JFSK15, FL19, FML15, LHFL19, NL20].

2.2. Deep Neural Networks for Visualization

Recently, many visualization approaches have been established that use deep neural networks to gain a better understanding of a dataset to be analyzed [WCWQ21]. Here, often an emphasis is put on the output of the networks. More precisely, the outputs of these networks represent already the final visualization. Generative networks like generative adversarial networks (GAN) or variational autoencoders make up the majority of the selected architectures [WITW20, HZCW21, HWG*19, LSS*19, WZH19]. Besides the direct generation of visualizations, some works also target the learned features within a network. Here, one exploits the property of deep neural networks that they learn a feature representation of the input after each layer. Deep autoencoders are networks that are specifically designed to encode an arbitrary input into a latent feature space representation, from which they can, in turn, decode the original input. Porter et al. [PXvO*19] perform a dimensionality reduction of the latent feature descriptors and select representative timesteps in the projected space for time-varying multivariate datasets. Sun et al. [SWG*19] and Han et al. [HTW18] use the latent feature space of autoencoders for clustering within spatio-temporal datasets. Jo et al. [JS19] use autoencoders to transform a $(64x64x1)$ scatter plot image to a $(32x1)$ feature representation to perform further in-depth analysis. Tkachev et al. [TFE21] developed an autoencoder-based approach that enables interactive example-based queries for similar behavior in ensembles of spatio-temporal data.

Transformer networks represent a network architecture that shows its strengths primarily in processing sequential inputs and outputs. Transformer-based architectures have shown state-of-the-art performances in several neural language processing [WDS*20] and computer vision [DBK*20] tasks. Thus, several visualization methods have been developed, which utilize transformer networks [NKWW21, JKV*21] with a particular focus on the analysis of the attention mechanism [Vig19, AZ20, CGW21].

All these works have in common that in order to create new visualizations, the proposed networks have to be trained for the specific dataset. They generalize for the given problem but have to be re-trained or transfer-learned for new kinds of data. SimilarityNet, in contrast, is trained once on a single dataset. Afterwards, it can be applied to unseen data of different application scenarios without re-training.

3. SimilarityNet

In this section, we describe SimilarityNet, a deep neural network with a latent feature space that captures the characteristics of temporal evolutions of spatio-temporal ensembles. It is trained with a generic phantom dataset, whose generation is detailed in Section 3.1. The deep neural network architecture is explained afterwards in Section 3.2, the training process in Section 3.3 and the obtained visualization in 3.4.

3.1. Training Data Generation

Spatio-temporal ensembles often consist of a large number of timesteps and a considerably high spatial resolution leading to a large number of spatial samples, especially when dealing with volume data. This leads to rather large datasets that usually cannot be fully stored in the system's working memory. For many algorithms and methods, this poses a problem, which is why various approaches have been developed to deal with the large memory overhead [HGSP18, FL19, HHB15]. One effective approach is Monte-Carlo sampling of individual volumes [FML15], which drastically reduces memory requirements while preserving information of the

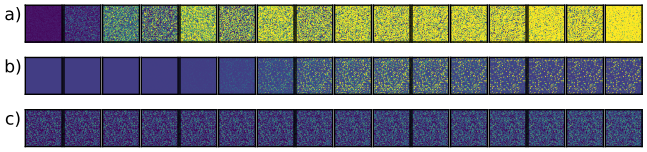

Figure 1: *Monte-Carlo samples of 16 timesteps of ensemble a) White dwarf, b) Semiconductor, c) Deep Water Asteroid (water) datasets. Further information about the datasets can be found in Section 4.*

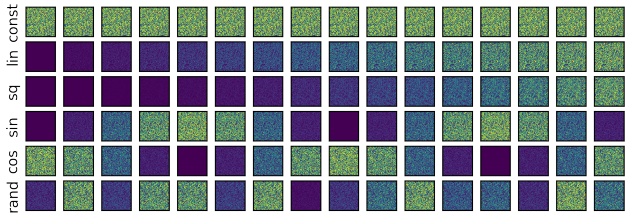

Figure 2: *Excerpt of the phantom dataset. Here we can see 6 different functions which are applied over time to a random distribution.*

volume with a negligible loss and adapting to any kind of spatial dimensionality. The underlying assumption is that the spatial variation of fields is continuous or even smooth, which is commonly true for physical simulations. Thus, sampling the space with a significantly lower amount of data samples suffices to capture the main spatial distribution, and the random character of Monte-Carlo approaches ensures that all spatial regions are captured [FML15].

Given a 4D ensemble with R simulation runs, each run $r \in \{0,\dots,R-1\}$ consisting of T_r timesteps t_r^n, $n \in \{0,\dots,T_r-1\}$, we perform a Monte-Carlo sampling by selecting k random spatial samples s_r^n from the given field with a consistent spatial distribution for all t_r^n. Thus, we use the same random sample locations for each volume of all timesteps of all simulation runs, which allows us to maintain spatial information and to perform a respective comparison between the fields. The higher the number of chosen samples is, the better the spatial information is captured. Examples of s_r^n for different timesteps t_r^n of a selected simulation run r for different datasets are shown in Figure 1. Although the intensities of the samples are obtained from the spatial position of the continuous or even smooth scalar fields, it appears that these distributions resembles random noise due to the random distribution of the spatial samples.

We take advantage of this characteristic by designing a dataset consisting of multiple time series with artificially generated noise. This dataset acts as a phantom dataset for Monte-Carlo sampled spatio-temporal ensembles and covers the same range of values as our original normalized samples. To create time series, we induce a temporal evolution on our artificially generated noise data by applying several temporal basis functions. An excerpt from this dataset can be found in Figure 2. Here, eight different temporal functions including constant (const), random (rand), sinus (sin), cosine (cos), linear (lin), and square (sq) functions for the temporal evolution are used. Inverse variants of linear and square functions are also present in this dataset but not shown in the figure. These basis functions are applied to the intensity of the random distribution over time. The basis functions were selected to represent common temporal behaviors of smooth physical phenomena. The figure shows time series for Gaussian noise distributions, but the complete dataset also holds the same variations for uniform random and Poisson distributions. In total, we generated a phantom dataset with 24 runs, each with up to 256 timesteps, and for each timestep we used $k = 16,384$ "spatial" samples.

3.2. Network Architecture Design

We have designed SimilarityNet to not only be able to accept arbitrarily many timesteps as input, but also to translate this input into a low-dimensional latent space representation. For this latent space projection we have chosen an autoencoder network architecture. Today, dimensionality reduction for data visualization is considered a typical application of autoencoders. With appropriate constraints on dimensionality, autoencoders can learn data projections that are more insightful than standard dimensionality reduction techniques such as prinicpal component analysis (PCA) [AEE17, CYLL18]. Autoencoders, also referred to as Siamese Neural Networks or U-Net, are automatically learned from data examples in an unsupervised fashion. That means that we can train specialized algorithm instances that work well on a particular input type. This does not require new engineering, only the appropriate choice of training data.

Figure 3 shows an overview of the architecture of SimilarityNet and illustrates the data pipeline. Our autoencoder consists of three main parts:

1. ε: $\mathbb{R}^k \Rightarrow \mathbb{R}^l$ (Encoder with dimensionality reduction)
2. τ: $\mathbb{R}^l \Rightarrow \mathbb{R}^l$ (Latent feature transformer processing)
3. δ: $\mathbb{R}^l \Rightarrow \mathbb{R}^k$ (Decoder)

Technically, the full encoder is made up by the functions ε and τ and the decoder of δ. The network's input is defined as $X^r = \{s_1^r, s_2^r, ..., s_n^r\}, s_i^r \in \mathbb{R}^k, i \in \{0,\dots,T_r\}, r \in \{0,\dots,R\}$, where s_i^r are samples as described in Section 3.1. The goal is to reconstruct a given input X^r, such that

$$\delta(\tau(\varepsilon(X^r))) = X^r. \tag{1}$$

The first part of SimilarityNet, ε, is a straight-forward funnel structure, where we reduce the input's dimensionality in every layer. Here, we select a projection dimensionality d, which defines the number of neurons in our first fully connected layer. Now, we consecutively reduce the dimensionality of the subsequent layers until we reach the desired latent space dimensionality l. Therefore we refer to $\varepsilon(X^r)$ as being our latent space representation.

The next part of SimilarityNet, τ, takes the latent space representation $\varepsilon(X^r)$ as an input and handles the sequential nature of our data. Here, we chose a transformer layer which processes $\varepsilon(X^r)$ before decoding. Self-attention architectures have recently shown breakthrough success in sequence-to-sequence tasks like natural language processing (NLP), achieving state-of-the-art results in

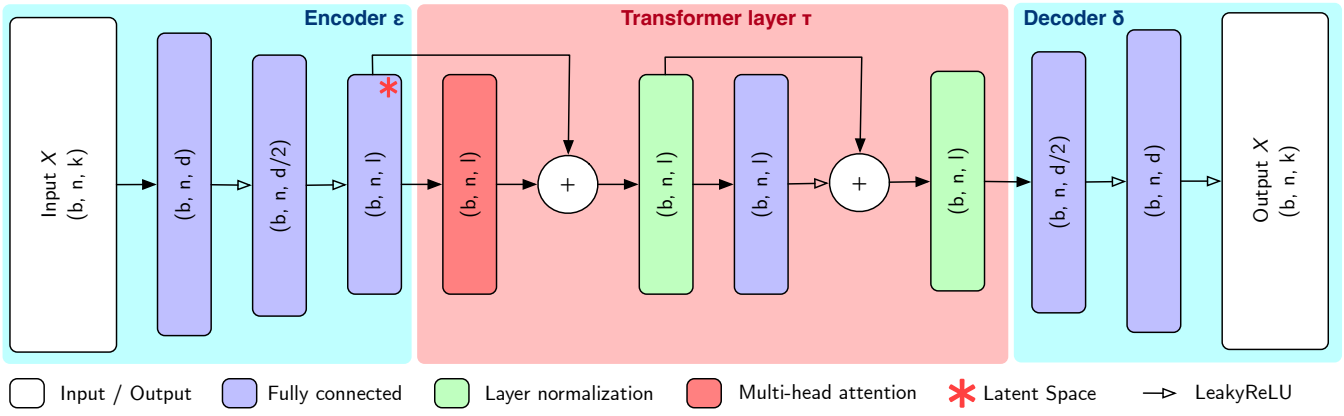

Figure 3: *Schematic depiction of SimilarityNet's architecture. All layers have been color-coded by it's type. The network is split into three parts: 1. the encoder* ε *which generates the latent feature representation; 2. the transformer layer* τ*, which processes the latent representation; 3. the decoder* δ *that is trained to decode the transformer layer's output back to the original input X. The output shape of all layers is also shown in every step (b: Batch size, k: Sampling dimensionality, n: Timestep count, d: Projection dimensionality, l: Latent space dimensionality).*

language modeling and machine translation [WDS*20, DBK*20]. They can effectively preserve information over long time horizons and scale to large amounts of data. Our chosen architecture for the transformer layer τ is derived from the transformer layer as proposed in [VSP*17]. In contrast to the transformer architecture in [VSP*17], positional encoding is not being used in SimilarityNet. This design choice makes it possible to enter sequences of arbitrary length into the model. The essential component of the transformer layer is the self-attention (or multi-head attention). The purpose of the self-attention is to calculate the correlation of an input symbol to all other input symbols. In our context, symbol stands for timestep samples. The self-attention algorithm uses three trained weight matrices to obtain the three entities query (q), key (k), and value (v). Query and key define the pairwise relationship between the elements of the sequence, and value establishes the context of the analyzed element. The three vectors q,k,v are calculated by multiplying $\varepsilon(X^r)$ with the matrices Q, K and V, which are learned by training, i.e.:

$$\begin{aligned} q &= \varepsilon(X^r)\cdot Q \\ k &= \varepsilon(X^r)\cdot K \\ v &= \varepsilon(X^r)\cdot V \end{aligned} \tag{2}$$

From these computations, the score s is calculated by

$$s = q\cdot k. \tag{3}$$

The Softmax function σ is then applied to s and multiplied by the value vector v. This leads to the result that symbols, which are not important for the semantic, are multiplied with a small value and symbols which are important for the semantic are multiplied with a high value [VSP*17]. The output of the attention module z is therefore defined as:

$$z = \sigma\left(\frac{s}{\sqrt{\dim(k)}}\right)\cdot v \tag{4}$$

where we divide by the square root of the length of the key-vectors $\dim(k)$ to guarantee more stable gradients.

After encoding the input $\tau(\varepsilon(X^r))$, we decode this representation to reconstruct X^r using δ. For this, analogous to ε, we have designed an inverted funnel that successively increases the dimensionality of the data from l to d to k.

We made several design iterations to determine a network architecture that generates the desired outputs with the fewest amount of network parameters. Here, we also tried to remove the transformer layer. The adjusted network then still produces outputs that are in many cases comparable to the transformer variant, but on closer examination we observed that, when not using the transformer layer, the network does not overfit as much as described in Section . In order to create comparable results without the transformer layer, we would have to increase the projection dimensionality d, which directly leads to a significantly higher number of trainable parameters in the network.

3.3. Training

In principle, SimilarityNet is trained like a standard autoencoder network. We input time series of spatial samples into the network and expect them to be reconstructed by the network. Given an input X^r, we minimize the mean-squared-error loss L given by

$$L(X^r) = E(||X^r - \delta(\tau(\varepsilon(X^r)))||^2).$$

Our training takes place merely on the phantom dataset described in Section 3.1. Here, the network is trained so that it can reconstruct the phantom data almost perfectly. In fact, the network is trained until the latent feature representations of the phantom data can be represented by a constant if possible (see Figure 4). The network is thus strongly overfitted to the training data.

By overfitting the network to the phantom dataset and learning constant latent outputs, any input to the network that deviates from

the training data generates latent outputs that diverge from the constants. The more training data have been overfitted and the more runs generate constant latent outputs, the better the difference to the training data is captured.

As soon as a new, unseen run X_i is being processed by SimilarityNet, we obtain $\varepsilon(X_i) = x_i$, a latent representation of each timestep of X. Ensembles are usually created repeatedly for one and the same phenomenon for different simulation parameters or initial setups. Despite their intended differences, there is a notable degree of similarity in the general structure and spatial layout across all simulation runs of the ensemble. Consequently, if another run X_j from the same ensemble is processed by SimilarityNet, then the outputs $\varepsilon(X_j) = x_j$ are correspondingly similar to x_i, yet distinguishable enough for an in-depth similarity analysis of the ensemble members and their evolution.

Our experiments show that these inter-run feature differences show strong similarities to other established embedding methods, which gives us the possibility to use SimilarityNet for a similarity analysis.

3.4. Latent Space Visualization

For the visualization of any (unseen) spatio-temporal ensemble, we first perform a Monte-Carlo sampling with the same spatial sampling pattern on 2D or 3D fields of each timestep and each simulation run. The simulation runs are then fed to SimilarityNet to create the latent space representation of each timestep for all runs. For its visualization, the latent space representation is plotted over time while connecting consecutive timesteps of each simulation run to form a line. The outcome is a *similarity plot*, where the x-axis represents the time dimension of the simulations, the y-axis represents the latent space, and each simulation run is shown as a piecewise linear curve over the time period spanned by the respective simulation run, see Figure 4. Hence, the visualizations depict the temporal evolution of the ensemble and its members, where similar ensemble members are expected to stay close together, while trends such as diverging or converging behaviors should be captured and recognizable. We refrain from labeling the y-axis with its scale, as the respective values are created by the network and do not represent interpretable numbers.

Due to the fact that the network is strongly overfitted to the phantom dataset to produce constant values for it, the latent dimensions result in an almost identical outcome. Thus, for the analysis we can restrict ourselves to one latent dimension by picking any of the two latent space dimensions. In the following results, we selected the first latent dimension for our visualizations leading to 1D embeddings.

4. Results

In this section we are going to test SimilarityNet on various spatio-temporal ensemble datasets to document its effectiveness in capturing similarities and temporal evolutions of the ensemble members. We compare the similarity plots generated by SimilarityNet with those obtained by dimensionality reduction methods using two different distance metrics.

4.1. Dimensionality Reduction

We will compare our method against two Multidimensional Scaling (MDS) projection approaches. Here, we use MDS with two different distance functions. We consider MDS to be the most suitable dimensionality reduction method, as it minimizes stress, i.e., the distances computed by the chosen distance functions are preserved as much as possible during projection.

The first distance function computes *field similarity* (FS) [FL19]. Assuming that the field is normalized to the unit interval $[0,1]$, the field similarity distance $D_{FS}(A,B)$ is defined as

$$D_{FS}(A,B) = 1 - \frac{\sum_{i=0}^{k-1}(1 - max(a_i,b_i))}{\sum_{i=0}^{k-1}(1 - min(a_i,b_i))} \tag{5}$$

where $A = (a_0,\ldots,a_{k-1})$ and $B = (b_0,\ldots,b_{k-1})$ are the vectors of field values at the k Monte-Carlo sample points. Fofonov and Linsen [FL19] showed in a comparative study that this field similarity has desirable properties for computing distances within spatio-temporal ensembles.

The second distance function is based on the Pearson correlation coefficient, to which we refer as *correlation similarity* (CS). Using the same notations as above, the Pearson correlation coefficient is defined as

$$r_{AB} = \frac{\sum_{i=0}^{k-1}(a_i - \bar{A}) \cdot (b_i - \bar{B})}{\sqrt{\sum_{i=0}^{k-1}(a_i - \bar{A})^2 \cdot \sum_{i=0}^{k-1}(b_i - \bar{B})^2}} \tag{6}$$

where $\bar{A}$ and $\bar{B}$ denote the mean values of A and B, respectively. In order to transform r_{AB} to a normalized distance $D_{CS}(A,B)$, we perform the operation

$$D_{CS}(A,B) = 1 - \frac{r_{AB} + 1}{2} \tag{7}$$

that assigns high correlated fields ($r_{AB} = 1$) a value of 0 and anti-correlated fields ($r_{AB} = -1$) a value of 1.

For all MDS plots, we use the first principal component and plot the 1D embeddings over time in the same way as we display our similarity plots, cf. Section 3.4.

4.2. Experimental Setup

To generate the following results, SimilarityNet was trained once on the phantom dataset and then applied to the different ensembles. In order to find the best size of our proposed architecture, we gradually decreased the layers' dimensionalities and observed the training outputs (see Figure 4). We aimed at getting as many constant outputs as possible while decreasing the number of neurons per layer. The architecture can be seen in Figure 3. For training, we use LeakyReLU activation after all fully connected layers. Throughout our experiments, we found that this choice of activation function produced the fastest and most stable trainings. We use the Adam optimizer with a variable learning rate of 0.01 (reduced by 0.1% after each epoch), $\beta_1 = 0.9$, $\beta_2 = 0.98$, and $\varepsilon = 1e-9$. Our chosen loss function is the mean-squared error. The training takes place in batches of 8 runs. In total, we trained the network for 500 epochs, which took approximately 1 second per epoch on a MacBook Pro Laptop from 2021. The inference of a single run with

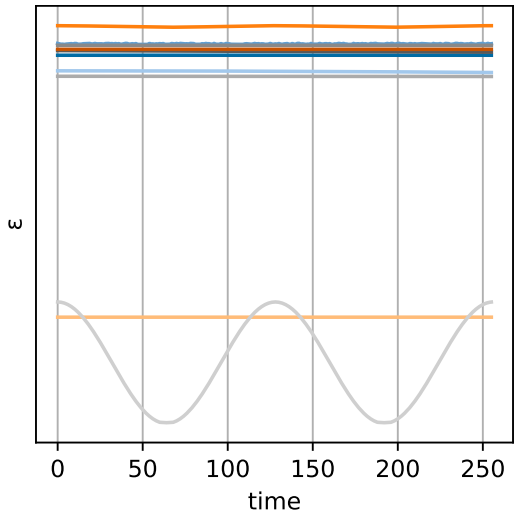

Figure 4: *Encoder ε of the phantom training data after 500 epochs of training. All variations except the cosinus basis function of a Poisson noise distribution (gray oscillating curve) are represented by a constant over time.*

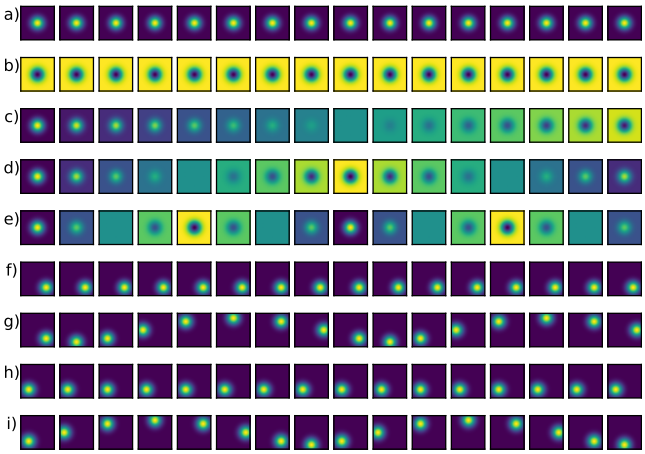

Figure 5: *Visualizations of the 2D fields for some selected timesteps (16 out of 256) of the synthetic dataset. Each run is normalized between 0 and 1 and accordingly color-coded from blue to yellow.*

about 256 timesteps takes 0.03 seconds on average. For the number of Monte-Carlo samples, we chose $k = 16,384$ samples [FML15]. We chose the projection dimension to be $d = 8$. Our latent space has a dimensionality of $l = 2$. We found this setup to be the best compromise between having a good reconstruction of the training data and network size. The resulting network has a size of approximately 1.7 Mb.

The resulting latent space visualizations of the phantom data can be seen in Figure 4. Here, all except one run have been encoded by a constant.

The SimilarityNet visualizations for all tested datasets (see below) use the same network with the same weights (i.e., the same training). To reduce clutter in the final similarity plots, when appropriate, we chose to plot each run in a separate row while each column (approach) shares the same y-range (e.g., Figure 6). If the clutter is negligible, we plot all runs in the same plot for each approach (e.g., Figure 4).

4.3. Synthetic Ensemble

First, we created a synthetic dataset that allows us to evaluate and compare the inter- and intra-run similarity intuitively. The chosen dataset consists of 256 timesteps with a spatial resolution of 256×256 and uses a 2D Gaussian distribution with a standard deviation $\sigma = 32$. An excerpt of this dataset can be seen in Figure 5. The first two runs, a) and b), are two constant runs, where the constant field in b) is the inverse of that in a). Runs c), d), and e) alternate at different frequencies between a) and b) using linear interpolation. Run f) and h) are again constant fields, but the center of the Gaussian kernel is positioned in the lower right and left of the field, respectively. Run g) and i) are runs that start at the same positions as f) and h), respectively, but rotate around the center twice.

We chose to analyze the synthetic dataset in two parts, a) - e) and f) - i), because the y-axis scaling of the two subsets differs significantly. Runs f) - i) have a much smaller latent feature value range than the interpolating runs, leading to a barely noticeable change over time when plotted in the same coordinate system.

Figure 6 shows the similarity plots of runs a) - e). Comparing all three methods, we can see that the constant runs a) and b) are plotted as a straight line for all three approaches, which is in line with our expectations. When using linear interpolations (c - e), we see apparent differences between the approaches. SimilarityNet and FS show a similar evolution, representing the continuously varying nature of the underlying data well. However, these two methods differ insofar as SimilarityNet shows an almost perfectly linear progression, while FS shows curves that resemble higher-order polynomials. Thus, we argue that SimilarityNet matches our expectations better, as the linear transitions lead to linear curves. CS, on the other hand, has abrupt jumps that do not reflect the nature of the data well.

Next we analyzed the runs f) - i) (see Figure 7). Here, a prominent resemblance between FS and CS is evident (mirrored on the y axis). In the SimilarityNet plots, however, a much stronger oscillation is noticeable. In the case of the rotating data, FS and CS approaches seem to reflect the nature of the given data better than SimilarityNet. Nevertheless, all plots highlight that a periodical pattern is taking place. Furthermore, it becomes clear that the runs g) and i) have the same course, but they have a different starting point and are therefore shifted in time.

4.4. Deep Water Asteroid Impact Ensemble

The first real-world dataset that we analyze is the Deep Water Asteroid Impact dataset [GHPW18]. In this ensemble, three simulation parameters (asteroid size, impact angle, airburst) have been varied to study the extent of an asteroid impacting deep-sea oceans. The spatial dimensionality of this dataset is $300 \times 300 \times 300$ and a

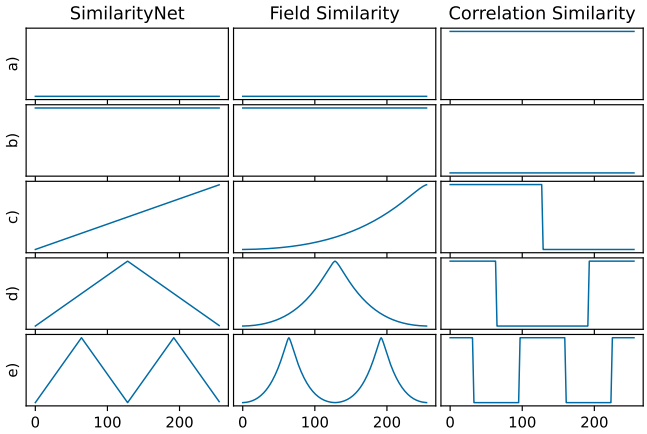

Figure 6: *Similarity plots of SimilarityNet compared to MDS using field and correlation similarity for the constant and piecewise linear runs of the synthetic dataset a) - e), see Figure 5. The x-axis depicts time, the y-axis the latent space or first principal component values. In order to reduce clutter, we chose to plot each run in a separate row. Each column shares the same y-range. The constant and piecewise linear behavior is perfectly captured using our SimilarityNet approach.*

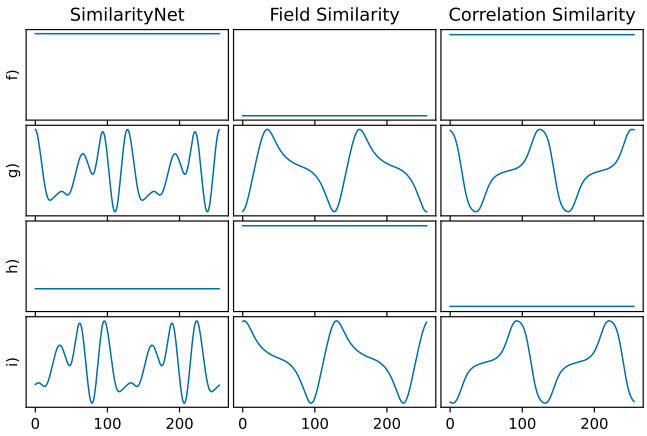

Figure 7: *Similarity plots of SimilarityNet compared to MDS using field and correlation similarity for the rotating and translated constant runs of the synthetic dataset f) - i), see Figure 5. The rotating behavior leads to oscillating patterns, where the two rotating runs g) and i) exhibit a time shift.*

single run can hold upto 487 timesteps. For our similarity analysis we make use of 4 scalar fields, temperature *tev*, pressure *prs*, volume fraction water *v02*, and volume fraction asteroid *v03*.

The similarity plots for this ensemble are shown in Figure 8. The plots for the temperature field show a very similar picture between the SimilarityNet and FS output. In both plots, three groups become apparent, G_0 = (yA31, yA32), G_1 = (yB31, yC31) and G_2 = (yA11, yB11, yC11) . Each group shares similar simulation parameters encoded in the runs' names (last three characters). In the CS plot, the groups are recognizable but not as prominent as in the

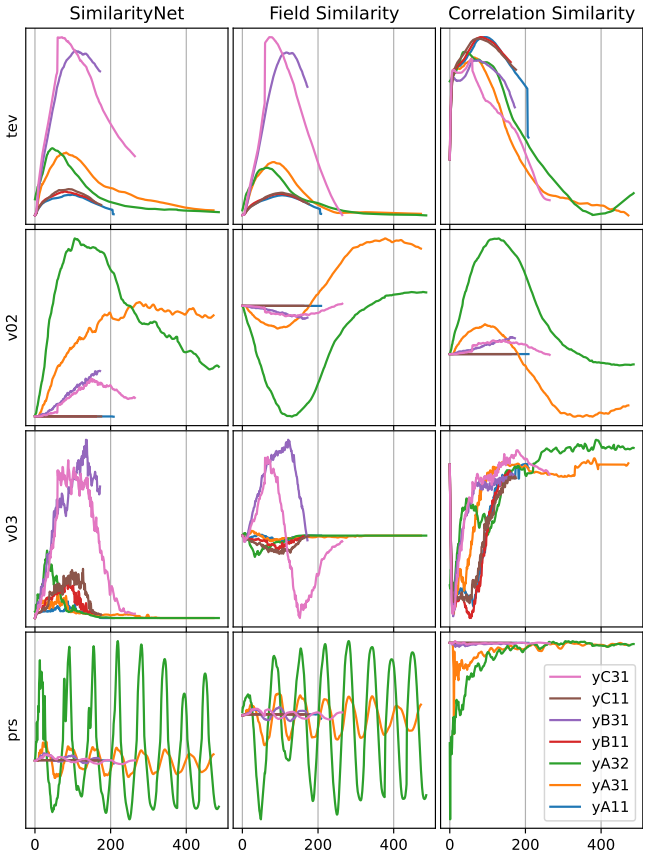

Figure 8: *Similarity plots of SimilarityNet compared to MDS using field and correlation similarity for Deep Water Asteroid Impact dataset. For each field (tev, v02, v03, prs) we can see the evolution of seven runs in all three similarity approaches. The runs' parameter configuration are encoded in their names. For each field, we can see groups with similar temporal evolution, which share common simulation parameters.*

other approaches. All plots show a rapid jump in the first quarter of the pink (yC31) and at the end of the blue run (yA11), which indicates missing or corrupted timesteps. Similar groupings also occur in the field volume fraction water (v02), but here, a bigger difference among G_0 runs become apparent in all three approaches. Investigating this further gave us the insight, that the voxel spacings of these two runs differ, which leads to a denser sampling in *yA*32 (smaller voxel spacing). We see a resemblance between the SimilarityNet and FS plots in the field volume fraction asteroid (v03). Especially the group G_1 stands out here. The pressure field shows an interesting phenomenon due to erroneous boundary conditions in the simulation. Here the pressure wave generated by the asteroid reflects at the boundaries of the bounding box. The strong sinusoidal signal indicates that a recurring pattern is taking place over time. This is especially evident in the SimilarityNet as well as in the FS plots. The CS plots also shows an oscillating pattern, but not as prominently as the other two approaches.

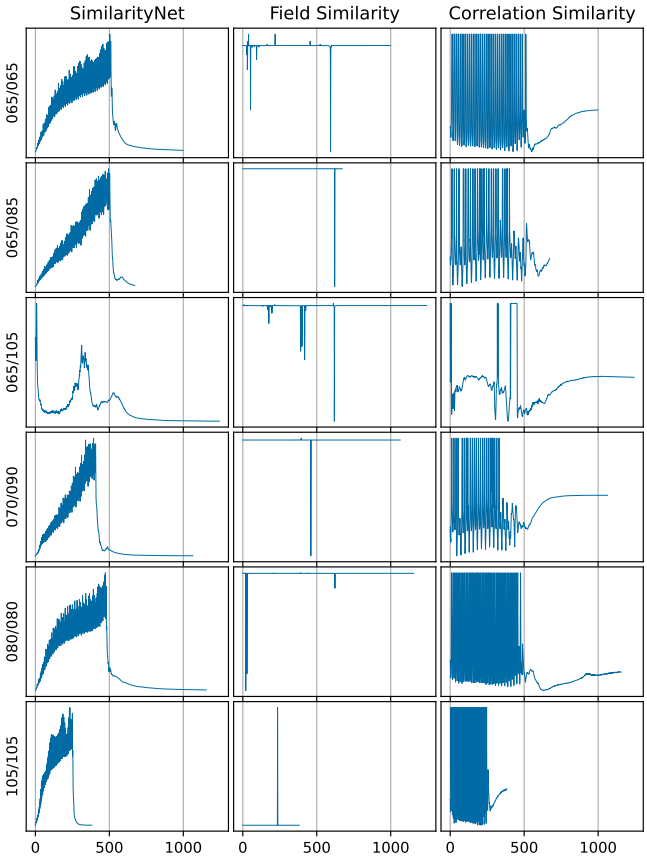

Figure 9: *Similarity plots of SimilarityNet compared to MDS using field and correlation similarity for White Dwarf dataset. We picked six representative runs. The name of each run (left axis) depicts the respective simulation parameters, namely the masses of both stars. All plots of SimilarityNet and correlation similarity, except for run 065/105, show an oscillating pattern which rapidly changes at around timestep 500 to a nearly constant curve progression. The field similarity approach suffers from very strong outlier-timesteps in all runs.*

4.5. White Dwarf Ensemble

The White Dwarf dataset simulates a binary star system where two stars rotate around each other at different masses until they finally merge and end in a large mass ejection. Here, the masses of the two stars are the varying simulation parameters. If the stars are of similar mass, they rotate longer than if there is a significant difference. In total, this ensemble consists of 42 runs with different mass combinations and different fields. We have picked six runs and the temperature field. We selected the masses to cover the possible simulation scenarios. In the similarity plots, which can be seen in Figure 9, we can observe a periodically repeating pattern. Here we can see a strong oscillation starting at the beginning of the runs, which is visible in the SimilarityNet and the CS plots. In direct comparison, the runs in the SimilarityNet plots also show an increasing behavior in addition to the oscillation. This is not present in the CS plot. The moment of the merging of the stars can also be seen in both plots. In both approaches, the oscillation disappears as soon as the stars have merged. For a large mass difference of the rotating stars (run 065/105), both similarity plots do not show a periodical pattern, because these stars merge immediately. When zooming into timespan $[480, 549]$ of run 065/065 and investigating volume renderings of selected timesteps (see Figure 10), we can observe that the rotational movement of the two objects is reflected by the oscillating pattern in the SimilarityNet plot. The merge event of the two objects can also be seen starting from timestep 520. At the same time, the oscillating pattern vanishes (compare Figure 9 and 10)

Note that the FS results suffer from very sharp timestep outliers, which influence the scaling so much that the plot cannot be analyzed meaningfully without further effort.

4.6. Max Planck Grand Ensemble

We also applied our approach to the Max Planck Institute Grand Ensemble [MMSG*19] dataset. This ensemble simulates the course of climate from 2006 until 2099 at a spatial resolution of 192×96. We used the mean monthly surface air temperature field from 3 randomly picked runs for our similarity analysis. Since the point-wise variance of monthly temperature data is dominated by the seasonal cycle, we derived anomalies with respect to the climatological mean monthly values of the period 2006-2015. Due to global warming, the overall trend of this seasonally adjusted dataset should result in an increased temperature over time for all runs.

When analyzing the similarity plots in Figure 11, we can immediately see that CS shows an unexpected and undesired behavior with repeated jumps to a specific value. In contrast, the SimilarityNet and FS plots exhibit the expected linear change over time, where the FS plot was flipped along the y-axis (the principal component is only unique up to its sign) for an easier comparison of the two plots. Here, the individual runs are barely distinguishable between SimilarityNet and FS. The course of the runs in comparison to each other is also almost identical in the two plots.

4.7. Semiconductor Ensemble

Finally, we apply the approaches to the Semiconductor dataset, where it is investigated how photons are emitted from a semiconductor quantum wire. This is a 2D dataset in phase space rather than physical space, but this does not impede the direct application of our methods. A total of four simulation parameters were varied here. The ensemble consists of 150 runs of 70 timesteps at a spatial resolution of 300×200 each. We randomly selected a subset of 10 runs.

When comparing each run embedding seperately as shown in Figure 12, we can see a comparable behavior for several runs' temporal progression, especially between SimilarityNet and FS (r00, r04, r06, r07, r08). CS created results, which do not compare to neither SimilarityNet or FS. The similarities between SimiarityNet and FS have to be put into perspective because clear differences become visible when all runs are plotted simultaneously (see Figure 13). Here, for example, the order of the shown runs (from top right to bottom) is not identical. Nevertheless, some groupings become apparent, which show up across all three approaches (e.g. red and

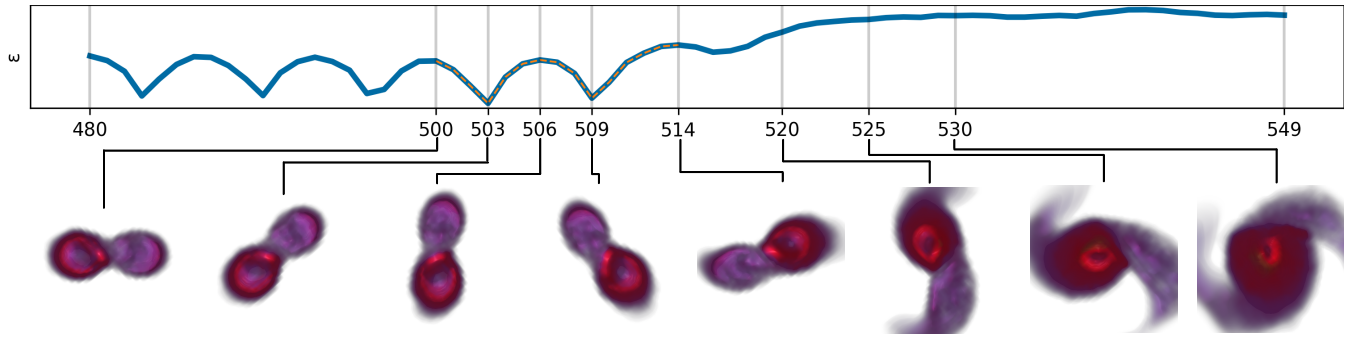

Figure 10: *SimilarityNet's encoder output* ε *for time range* $[480, 549]$ *of ensemble run 065/065 (White Dwarf dataset, compare Section 4.5 and Figure 9). Volume renderings are shown for selected timesteps. The transfer function has been manually adjusted to highlight the run's main features. A half rotation of the two stars is observed in timesteps 500 to 514 (orange dashed line). Starting from timestep 520, a merge event of the two stars can be observed in the volume renderings, which coincides with the disappearance of the oscillating pattern in the embedding.*

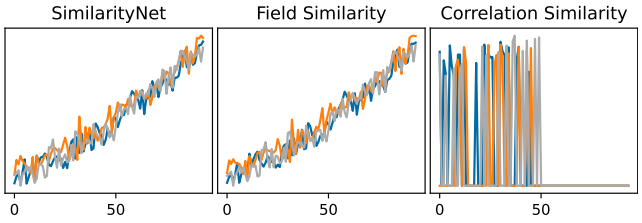

Figure 11: *Similarity plots of SimilarityNet compared to MDS using field and correlation similarity for seasonally adjusted Max Planck Grand Ensemble dataset (sea surface temperature field). All plots depict the temporal progression of three randomly picked runs. SimilarityNet and field similarity show a linear change over time. Correlation similarity suffers from outlier-timesteps.*

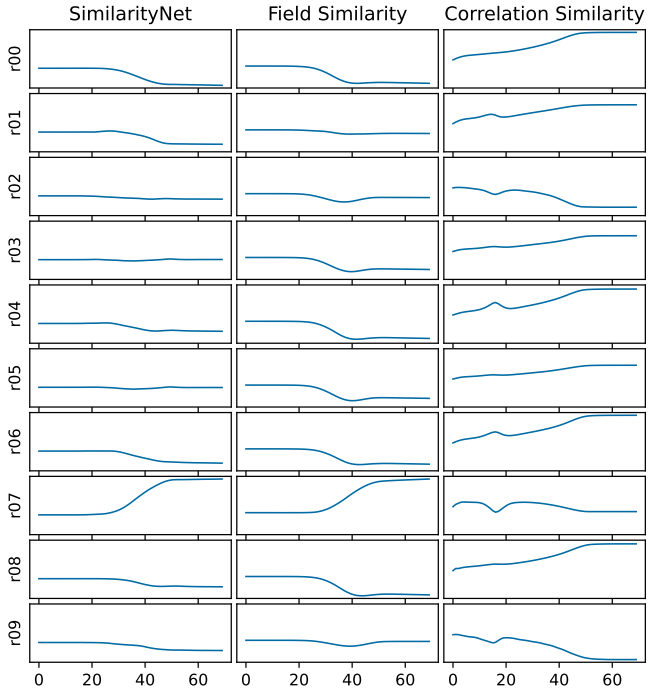

Figure 12: *Similarity plots of SimilarityNet compared to MDS using field and correlation similarity for Semiconductor dataset. We randomly select 10 runs and juxtapose them for each similarity approach.*

black run). However, when analyzing this dataset, the fact that we are only looking at the first principal component of FS and CS can also lead to the observed differences in the similarity plots.

5. Discussion and Conclusion

We presented SimilarityNet, a deep neural network architecture coupled with a novel training strategy that is designed to generate latent feature space representations which can be utilized for spatio-temporal ensemble similarity analysis. By analyzing five different datasets we have shown that the similarity plots generated by SimilarityNet can be interpreted as a representation of the temporal progression of ensemble runs. Not only is the evolution within a run well represented, but the comparison across runs can also be made intuitively. While being able to generate reasonable similarity plots for various kinds of ensemble datasets, SimilarityNet has never been trained on any of those. The training strategy as proposed in Section 3.3 results in a network state, that is able to process unseen data in a way, that the differences to the originally trained data are encoded in the latent space. The phantom dataset used for training represents a number of functions over time that serve as a basis for handling unseen data.

The results generated by SimilarityNet often show significant similarities to the field similarity MDS projections (see Section 4.3, 4.4), sometimes this is even exceptionally high (see Figure 11). When testing SimilarityNet on our synthetic dataset it generated more intuitive results for piecewise linear evolutions than the MDS plots. However, this does not apply to rotating behaviors. Throughout our experiments, SimilarityNet was able to reliably generate

Ensemble	Run count	Duration Field Similarity [s]	Duration Correlation Similarity [s]	Duration SimilarityNet [s]
Synthetic	9	372	452	0.27
Deep Water Asteroid Impact	7	498	546	0.22
White Dwarf	6	4080	2402	0.36
Max Planck Grand Ensemble	3	520	695	0.10
Semiconductor	10	68	85	0.17

Table 1: *Table of all tested ensembles. For each ensemble, we list how many runs we analyze and how long the calculations for each similarity approach take.*

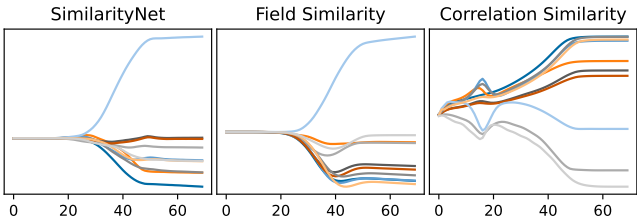

Figure 13: *The same similarity plots as in Figure 12, but here we show all runs in a single plot for each similarity approach.*

embeddings for all of our tested datasets. The MDS approaches sometimes failed to do so (see Figures 9 and 11), as the distance measures are not equally suitable for all data sets with different data characteristics such as containing outliers.

As SimilarityNet can directly be applied to any unseen spatio-temporal ensemble, it can reliably be applied to a large number of arbitrary ensembles without long computation times. In fact, it outperforms the combination of distance matrix computations and embedding computations by approximately three order of magnitude for the examples presented here, see Table 4.7. The computation time of SimilarityNet increases linearly with the number of timesteps/runs per ensemble because each run can be inputted individually to the network. As a consequence, the memory requirement of SimilarityNet also increases linearly with the number of timesteps per run. In contrast, the computation time and memory requirements for distance-based approaches (e.g., MDS) grow quadratically with the number of considered timesteps. Thus, huge ensembles cannot be computed as a whole by these approaches on commodity computers.

In this work we focused on analyzing spatio-temporal data, therefore we designed the network such that its input can be a time-series of high-dimensional vectors. However, one can easily use the same network to create the 1D embeddings of static (i.e., not time-dependent) datasets.

We want to emphasize that the objective of our work is not to generate quantitatively better embeddings than the two methods we compare to. Rather, we want to show that our visualizations are able to produce comparable results that allow for a similar analysis of the 1D embeddings while benefiting from significantly lower computational costs. Moreover, our method does not require us to choose a distance metric.

In this work, we mainly focused on 1D projections. Currently, we set our latent space to be two-dimensional, but the two learned features do not differ noticeably, i.e., they do not allow for a 2D similarity analysis like the ones known from 2D embeddings obtained by classical dimensionality reduction techniques. Our goal was to produce 1D embeddings and not to detect multiple principal components as in MDS approaches. Hence, we can also not analyze the strengths of the principal components. In future work, we plan to address this by introducing activation regularizations for the latent features that are designed to produce orthogonal outputs by penalizing correlating features. Our visual representations of simulation runs as curves over time scales reasonably well with the number of simulation runs. Of course, at some point an increased number of runs will lead to visual clutter such that interaction mechanisms like filtering should be used.

In future work, we will also focus our analysis on the attention mechanism introduced by SimilarityNet's transformer layer. Here, we have good insights indicating that the attention mechanism can eventually be used for key-frame detection. Another possible use of SimilarityNet are interactive real-time applications, where a user wants to get immediate feedback about the similarity of a selected ensemble.

The architecture, as well as notebooks for the training data generation, can be found on http://github.com/khuesmann/similarity-net.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) grant 260446826 (LI 1530/21-2).

References

[AEE17] ALMOTIRI J., ELLEITHY K., ELLEITHY A.: Comparison of autoencoder and principal component analysis followed by neural network for e-learning using handwritten recognition. In *2017 IEEE Long Island Systems, Applications and Technology Conference (LISAT)* (2017), IEEE, pp. 1–5. 3

[AZ20] ABNAR S., ZUIDEMA W.: Quantifying attention flow in transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020), pp. 4190–4197. 2

[BPFG11] BERGER W., PIRINGER H., FILZMOSER P., GRÖLLER E.: Uncertainty-aware exploration of continuous parameter spaces using multivariate prediction. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 911–920. 2

[CGW21] CHEFER H., GUR S., WOLF L.: Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 782–791. 2

[CYLL18] CHEN Z., YEO C. K., LEE B. S., LAU C. T.: Autoencoder-based network anomaly detection. In *2018 Wireless Telecommunications Symposium (WTS)* (2018), IEEE, pp. 1–5. 3

[DBK*20] DOSOVITSKIY A., BEYER L., KOLESNIKOV A., WEISSENBORN D., ZHAI X., UNTERTHINER T., DEHGHANI M., MINDERER M., HEIGOLD G., GELLY S., ET AL.: An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations* (2020). 2, 4

[EHNP04] EDELSBRUNNER H., HARER J., NATARAJAN V., PASCUCCI V.: Local and global comparison of continuous functions. In *IEEE Visualization 2004* (2004), IEEE, pp. 275–280. 1

[FFH21] FRIEDERICI A., FALK M., HOTZ I.: A winding angle framework for tracking and exploring eddy transport in oceanic ensemble simulations. *The Eurographics Association* (2021). 2

[FJC*19] FRANCH G., JURMAN G., COVIELLO L., PENDESINI M., FURLANELLO C.: Mass-umap: Fast and accurate analog ensemble search in weather radar archives. *Remote Sensing 11*, 24 (2019), 2922. 2

[FL19] FOFONOV A., LINSEN L.: Projected field similarity for comparative visualization of multi-run multi-field time-varying spatial data. In *Computer Graphics Forum* (2019), vol. 38, Wiley Online Library, pp. 286–299. 1, 2, 5

[FML15] FOFONOV A., MOLCHANOV V., LINSEN L.: Visual analysis of multi-run spatio-temporal simulations using isocontour similarity for projected views. *IEEE transactions on visualization and computer graphics 22*, 8 (2015), 2037–2050. 1, 2, 3, 6

[GHPW18] GISLER G. R., HEBERLING T., PLESKO C. S., WEAVER R. P.: Three-dimensional simulations of oblique asteroid impacts into water. *Journal of Space Safety Engineering 5*, 2 (2018), 106–114. 6

[Ham94] HAMBY D. M.: A review of techniques for parameter sensitivity analysis of environmental models. *Environmental monitoring and assessment 32*, 2 (1994), 135–154. 2

[HGSP18] HE W., GUO H., SHEN H.-W., PETERKA T.: efesta: Ensemble feature exploration with surface density estimates. *IEEE transactions on visualization and computer graphics 26*, 4 (2018), 1716–1731. 2

[HHB15] HAO L., HEALEY C. G., BASS S. A.: Effective visualization of temporal ensembles. *IEEE Transactions on Visualization and Computer Graphics 22*, 1 (2015), 787–796. 2

[HTW18] HAN J., TAO J., WANG C.: Flownet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE transactions on visualization and computer graphics 26*, 4 (2018), 1732–1744. 1, 2

[HWG*19] HE W., WANG J., GUO H., WANG K.-C., SHEN H.-W., RAJ M., NASHED Y. S., PETERKA T.: Insitunet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE transactions on visualization and computer graphics 26*, 1 (2019), 23–33. 2

[HZCW21] HAN J., ZHENG H., CHEN D. Z., WANG C.: Stnet: An end-to-end generative framework for synthesizing spatiotemporal super-resolution volumes. *IEEE Transactions on Visualization and Computer Graphics* (2021). 2

[JFSK15] JÄCKLE D., FISCHER F., SCHRECK T., KEIM D. A.: Temporal mds plots for analysis of multivariate data. *IEEE transactions on visualization and computer graphics 22*, 1 (2015), 141–150. 2

[JKV*21] JAUNET T., KERVADEC C., VUILLEMOT R., ANTIPOV G., BACCOUCHE M., WOLF C.: Visqa: X-raying vision and language reasoning in transformers. *IEEE Transactions on Visualization and Computer Graphics* (2021). 2

[JS19] JO J., SEO J.: Disentangled representation of data distributions in scatterplots. In *2019 IEEE Visualization Conference (VIS)* (2019), IEEE, pp. 136–140. 2

[KSHW21] KUMPF A., STUMPFEGGER J., HARTL P. F., WESTERMANN R.: Visual analysis of multi-parameter distributions across ensembles of 3d fields. *IEEE Transactions on Visualization and Computer Graphics* (2021). 2

[LHFL19] LEISTIKOW S., HUESMANN K., FOFONOV A., LINSEN L.: Aggregated ensemble views for deep water asteroid impact simulations. *IEEE computer graphics and applications 40*, 1 (2019), 72–81. 2

[LSS*19] LOMBARDI S., SIMON T., SARAGIH J., SCHWARTZ G., LEHRMANN A., SHEIKH Y.: Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751* (2019). 2

[MMSG*19] MAHER N., MILINSKI S., SUAREZ-GUTIERREZ L., BOTZET M., DOBRYNIN M., KORNBLUEH L., KRÖGER J., TAKANO Y., GHOSH R., HEDEMANN C., ET AL.: The max planck institute grand ensemble: enabling the exploration of climate system variability. *Journal of Advances in Modeling Earth Systems 11*, 7 (2019), 2050–2069. 8

[NKWW21] NARECHANIA A., KARDUNI A., WESSLEN R., WALL E.: Vitality: Promoting serendipitous discovery of academic literature with transformers & visual analytics. *IEEE Transactions on Visualization and Computer Graphics* (2021). 2

[NL20] NGO Q. Q., LINSEN L.: Interactive generation of 1d embeddings from 2d multi-dimensional data projections. 2

[NNN11] NAGARAJ S., NATARAJAN V., NANJUNDIAH R. S.: A gradient-based comparison measure for visual analysis of multifield data. In *Computer Graphics Forum* (2011), vol. 30, Wiley Online Library, pp. 1101–1110. 1

[PXvO*19] PORTER W. P., XING Y., VON OHLEN B. R., HAN J., WANG C.: A deep learning approach to selecting representative time steps for time-varying multivariate data. In *2019 IEEE Visualization Conference (VIS)* (2019), IEEE, pp. 1–5. 2

[STS06] SAUBER N., THEISEL H., SEIDEL H.-P.: Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 917–924. 1

[SWG*19] SUN J., WU C., GE Y., LI Y., YU H.: Spatial-temporal scientific data clustering via deep convolutional neural network. In *2019 IEEE International Conference on Big Data (Big Data)* (2019), IEEE, pp. 3424–3429. 1, 2

[TFE21] TKACHEV G., FREY S., ERTL T.: S4: Self-supervised learning of spatiotemporal similarity. *IEEE Transactions on Visualization and Computer Graphics* (2021). 1, 2

[Vig19] VIG J.: A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (2019), pp. 37–42. 2

[VSP*17] VASWANI A., SHAZEER N., PARMAR N., USZKOREIT J., JONES L., GOMEZ A. N., KAISER Ł., POLOSUKHIN I.: Attention is all you need. In *Advances in neural information processing systems* (2017), pp. 5998–6008. 4

[WCWQ21] WANG Q., CHEN Z., WANG Y., QU H.: A survey on ml4vis: Applying machinelearning advances to data visualization. *IEEE Transactions on Visualization & Computer Graphics*, 01 (2021), 1–1. 2

[WDS*20] WOLF T., DEBUT L., SANH V., CHAUMOND J., DELANGUE C., MOI A., CISTAC P., RAULT T., LOUF R., FUNTOWICZ M., ET AL.: Transformers: State-of-the-art natural language processing. In *EMNLP (Demos)* (2020). 2, 4

[WHLS19] WANG J., HAZARIKA S., LI C., SHEN H.-W.: Visualization and visual analysis of ensemble data: A survey. *IEEE transactions on visualization and computer graphics 25*, 9 (2019), 2853–2872. 2

[WITW20] WEISS S., IŞIK M., THIES J., WESTERMANN R.: Learning adaptive sampling and reconstruction for volume visualization. *IEEE Transactions on Visualization and Computer Graphics* (2020). 2

[WZH19] WANG Y., ZHONG Z., HUA J.: Deeporgannet: On-the-fly reconstruction and visualization of 3d/4d lung models from single-view projections by deep deformation network. *IEEE transactions on visualization and computer graphics 26*, 1 (2019), 960–970. 2