Local Prediction Models for Spatiotemporal Volume Visualization

Gleb Tkachev, Steffen Frey, Thomas Ertl

Abstract—We present a machine learning-based approach for detecting and visualizing complex behavior in spatiotemporal volumes. For this, we train models to predict future data values at a given position based on the past values in its neighborhood, capturing common temporal behavior in the data. We then evaluate the model's prediction on the same data. High prediction error means that the local behavior was too complex, unique or uncertain to be accurately captured during training, indicating spatiotemporal regions with interesting behavior. By training several models of varying capacity, we are able to detect spatiotemporal regions of various complexities. We aggregate the obtained prediction errors into a time series or spatial volumes and visualize them together to highlight regions of unpredictable behavior and how they differ between the models. We demonstrate two further volumetric applications: adaptive timestep selection and analysis of ensemble dissimilarity. We apply our technique to datasets from multiple application domains and demonstrate that we are able to produce meaningful results while making minimal assumptions about the underlying data.

Index Terms—Volume Visualization, Machine Learning, Neural Nets, Ensemble Visualization

1 Introduction

Increasingly fast computing systems as well as high-accuracy measurement techniques enable generation of spatiotemporal datasets with high resolution in both time and space. To analyze this kind of data, the most popular approach is still to manually browse through the timesteps individually or to use animations. This allows to look at the full spatial information, and well-known rendering and interaction techniques can be used. However, relying on animation alone has been shown to be ineffective, as only a limited number of frames can be memorized by an observer (e.g., [1]). Also, smaller details that can often be crucial can easily be missed. However, generating a meaningful static or summary visualization for spatiotemporal data is challenging due to issues ranging from occlusion and visual clutter to performance limitations for large data. For instance, selecting only a subset of timesteps for visualization (e.g. [2], [3], [4]) requires either data-specific selection criteria or involves the costly explicit quantification of timestep differences, and interesting process transitions may still be missed. On the other hand, while temporally dense techniques visualize all timesteps, these typically restrict themselves to specific (user-defined) characteristics or features to circumvent the occlusion problem ([1], [5], [6]).

In this paper, we integrate approaches from scientific visualization and machine learning and devise a new technique which allows us to construct models for detecting irregular processes in data without making any domain-specific assumptions.

After reviewing related work in Sec. 2, we discuss what we consider the main contributions of our work. *(I)* We introduce our prediction-based analysis method in Sec. 3 and *(II)* investigate potential causes of prediction errors in spatiotemporal data (Sec. 4). On this basis, *(III)* we present how to use the prediction results in visualization in Sec. 5. Furthermore, we present two applications building on top of the local prediction models: *(IV)* adaptive timestep selection (Sec. 6.1) and *(V)* an ensemble dissimilarity measure (Sec. 6.2). We describe our implementation in Sec. 7 and present the results in Sec. 8. Based on the results, *(VI)* we develop a semi-automatic method for parameter selection in Sec. 8.5. Finally, we discuss the current limitations and the future directions of our work in Sec. 9.

• *G.Tkachev, S.Frey and T.Ertl are with the Visualization Research Center of the University of Stuttgart. Email: {gleb.tkachev, steffen.frey, thomas.ertl}@visus.uni-stuttgart.de*

2 Related Work

Time-varying data visualization. A large body of work in time-dependent volume visualization is based on feature extraction. Time Activity Curves that contain each voxel's time series have been used in several techniques (e.g. [7], [8]). Lee et al. [9] extract trend relationships among variables for multifield time varying data. Wang et al. [10] extract a feature histogram per volume block, characterize the local temporal behavior, and classify them via k-means clustering. Based on similarity matrices, Frey et al. [11] detect and explore similarity in the temporal variation of field data. We also perform temporal feature extraction, however, we do not manually define our features, but learn them from the data.

Dutta and Shen [12] use Gaussian Mixture Models for tracking user-defined distribution-based features in volume data. Tzeng and Ma [13] use neural networks to generate adaptive transfer functions based on key frames. In contrast, our prediction models are unsupervised. Muelder and Ma [14] also use prediction, utilizing analytical predictor-corrector approach to track feature regions. However, we learn prediction mappings from the data and perform prediction directly on data values, without prior feature or predictor definitions.

Another line of work uses the notion of a space-time hypercube to apply operations like temporal transfer functions [5] or slicing and projection techniques [15] (see Bach et al. [16] for an overview). Tong et al. [3] use different metrics to compute the distance between datasets, and employ dynamic programming to select the most interesting timesteps on this basis. Based on a similar concept, Frey and Ertl [4], [17] generate a distribution-based distance to select timesteps. In a follow-up work, Frey [18] uses neural networks to estimate such distance metrics for time series data. Our

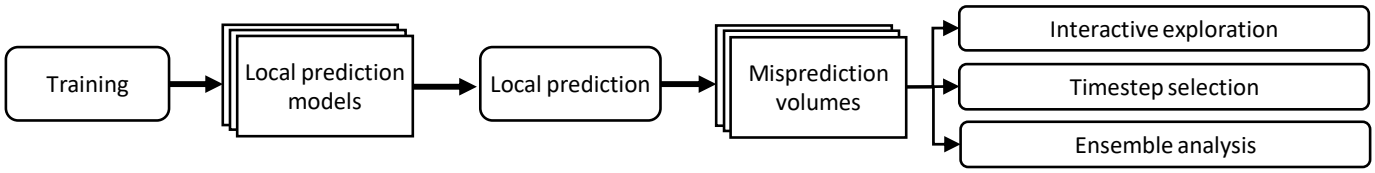


Fig. 1: Overview of our approach. We begin by training several local prediction models on a dataset. Then we perform prediction for the same data and obtain spatiotemporal misprediction volumes (error). We use the results for several applications, including exploration via spatial and temporal views, adaptive timestep selection and ensemble dissimilarity analysis.

technique also allows for adaptive timestep selection, but it relies on prediction-based temporal irregularity rather than distribution-based distance metrics.

Information theory in visualization Information theory has recently been gaining attention in visualization research. Bordoloi and Shen [19] use an entropy-based feature for view selection in volume rendering. Viola et al. [20] present an approach to automatically focus on objects in volumetric data by minimizing mutual information. Chen and Jänicke [21] as well as Wang and Shen [22] provide an overview and discuss the applicability of information theory in visualization. Related to our approach, Jänicke et al. [23] use the domain-agnostic information-theoretic notion of statistical complexity that estimates predictability of spatiotemporal regions. As opposed to using a probabilistic definition of predictability, we train an actual predictor on the data, measuring the error of the deviation as an estimate of irregularity. This allows us to not only benefit from higher-level features and generalization behavior of ML-models, but also to use multiple different models to produce a diverse visualization.

Machine learning in visualization In general, learning-based approaches are considered to have great potential in visualization (e.g. [24]). Apart from their application to specific tasks in time-varying data visualization, they have been of particular interest in the area of Visual Analytics (VA, see [25] for an overview). Fuchs et al. [26] demonstrate how to foster interaction between a human analyst and a genetic learning algorithm. In epidemiological analysis, Klemm et al. [27] employ decision trees to study relationships between image shape descriptors and non-image features.

While we use feed-forward neural networks in this work, unsupervised learning via self-organizing maps (SOMs) [28] has also gained popularity. For instance, Andrienko et al. [29] investigated how SOMs can be integrated into the visual analysis process of spatiotemporal data. Sacha et al. [30] presented a VA approach to analyze time series data using SOMs. Recently, neural networks were applied in the context of volumetric data. Berger et al. [31] used a neural network model with adversarial loss to generate and explore volume rendering images. And Han et al. [32] developed an autoencoder-based method of selecting and analyzing salient streamline and stream surfaces. We also employ neural networks for volume data, but we focus on the detection of spatiotemporal irregularities.

A few related approaches were investigated in the field of video analysis (preliminary results in [33], [34]), where neural networks are used to classify video frames into normal frames and shot boundaries. We also utilize machine learning models to detect events but for spatiotemporal visualization. More importantly, we do not make any assumptions about the events in the data, using prediction error as an indicator of irregular behavior, taking an unsupervised approach (as opposed to training the model to explicitly classify behavior as regular/irregular on supervised data).

3 Prediction-based irregularity analysis

Our objective is to detect and investigate irregular events in spatiotemporal volumes without making domain- or dataset-specific assumptions. For this, we analyze local temporal behavior using generic machine learning models, detecting regions of irregularity. In this section, we outline our approach in Sec. 3.1, before discussing the models (Sec. 3.2), our prediction problem formulation (Sec. 3.3) and how multiple models can be used to improve the analysis (Sec.3.4).

3.1 Concept and Outline

An overview of our approach is presented in Fig. 1. Specifically, we train a set of models to predict future data values based on the past. Once our models are trained, we evaluate them on the data and compare their prediction to the actual data. The prediction difference that we obtain is itself a spatiotemporal volume, which we use as an indicator of irregular behavior. Futhermore, we use multiple models of different capacity as detectors of different "sensitivity". Simpler models can only predict the most basic behavior and fail often, while more complicated models produce more compact and sparse regions of inaccurate prediction.

High prediction error in a particular region tells us that the model failed to capture the local behavior. There are several reasons why this could happen: the behavior is too complex to be expressed by the model; the behavior is rarely observed in the dataset (an outlier); available information is insufficient to predict the outcome. This is discussed in more detail in Sec. 4. We argue that these scenarios describe events of importance to the analyst and can be used to guide the visual exploration process, highlighting interesting events and providing a meaningful overview of the dataset, acting as a starting point for a more detailed and specialized analysis.

3.2 Prediction model

In principle, any model can be used as a predictor. However, accuracy of the prediction plays an important role: if the model cannot capture even trivial local behavior, we gain no additional information from the analysis. For simpler datasets, analytical models can be used, repeating past values or estimating local derivatives to do extrapolation. But as the data grows more complex, we require increasingly complex models, which inevitably implies additional assumptions and specialization. In contrast, we want our models to be uniformly applicable to datasets from different domains, and thus take a data-driven approach, using machine learning methods to learn local behavior from the data.

Specifically, we opted to use neural networks with simple architectures, utilizing only convolutional and densely-connected

layers. This provides us with a number of advantages. First of all, neural networks can represent a large number of functions and can be efficiently trained on large data. Second, simpler neural networks have fewer hyper-parameters and allow for easy and gradual adjustment of the model capacity, by changing the number of layers and neurons. More advanced network architectures, e.g. using recurrent connections, often require adjusting several related components together, each of which may have a complex impact on the prediction. Simple networks allow us to reason about the models in terms of their overall capacity, rather than qualitatively different architectures. Most importantly, by using basic densely-connected and convolutional layers we make as few dataset- or domain-specific assumptions as possible. Convolutional layers are basic locally-connected layers that exploit spatial coherence, which is a reasonable assumption for most types of scientific data. Nevertheless, we use networks both with and without convolutions, demonstrating their appropriateness for the task.

3.3 Local prediction problem

One of the key characteristics of our method is our local approach to prediction, i.e. the model predicts each future value based on values that occurred in the spatial neighborhood of the point in preceding timesteps. In principle, several other approaches could be taken, for example predicting the full field belonging to a timestep. However, there are a number of advantages to the local approach. Most importantly, locality removes positional information from the input data, making the learned prediction mapping translation-invariant. First, this makes the results more intuitive: equivalent behavior occurring in different locations in the data produces an equivalent effect on the prediction error, and thus has the same impact with respect to the visualization. Second, it simplifies the prediction problem, which means that much simpler models could be used for analysis: the model does not need to learn the same behavior in different locations separately. The local assumption also makes it less likely that the model will simply "memorize" the dataset, overfitting it heavily. A simple example would be an object that spontaneously appears: a local model has no way of anticipating that event. It has to predict that empty space leads to empty space, since that is what happens 99% of the time. If the model had full positional information, it could associate certain "landmarks" from distant regions of the data with the object appearing, effectively overfitting the dataset, requiring a lot of care with model regularization and validation. Local prediction also provides several performance gains. Using smaller patches and simpler models means that we require less training data and update fewer parameters, speeding up the training process. Finally, finer data partitioning increases the parallelization potential both during training and prediction. For further discussion see Sec. 9.

In formal terms, local prediction means that our model, given a voxel value $D(p,t)$ at the spatial position $p = (x,y,z)$ in timestep t, is trained to perform the following mapping:

$$Patch(p,t,l_s,l_t) \rightarrow D(p,t+1) \tag{1}$$

where $Patch(p,t,l_s,l_t)$ is a spatiotemporal box with spatial extent l_s and temporal extent l_t centered in space around point p. Note, that unlike the spatial extent l_s, the temporal extent l_t covers only one direction, thus the input includes only the past values, and none from the future.

For additional control over the difficulty of prediction, we generalize the problem to predicting not one, but d timesteps ahead. Thus, we add a delay between the input patch and the target value:

$$Patch(p,t,l_s,l_t) \rightarrow D(p,t+d), \quad d \geq 1 \tag{2}$$

This extension is useful for datasets with high temporal resolution. Increasing the value of d makes learning of simple mappings (e.g. repeating the most recent value) less feasible, forcing the model to learn more complex temporal relationships (detailed discussion in Sec. 8.4.1).

To obtain the data for training the model, we extract all possible spatiotemporal patches of spatial radius l_s, temporal extent l_t and delay d. This means, that for a dataset of size (X,Y,Z,T) we have n data points:

$$n = (X - l_s) \times (Y - l_s) \times (Z - l_s) \times (T - (l_t + d)) \tag{3}$$

Since the size of the extracted data grows polynomially with the dataset resolution, for large volumes it may easily reach terabytes, making the training computationally infeasible. To alleviate the problem, we perform random undersampling of the data, including a given patch into the training data with probability p_u.

After the training, we evaluate the model on the whole dataset, i.e. on every possible patch, obtaining a full spatiotemporal prediction volume. The prediction volume is slightly smaller than the original dataset, since we cannot perform prediction near the borders. Finally, we compute the absolute difference between the prediction volume and the original data. We use the resulting misprediction volume for visualization (e.g., Sec. 5).

3.4 Multiple prediction models

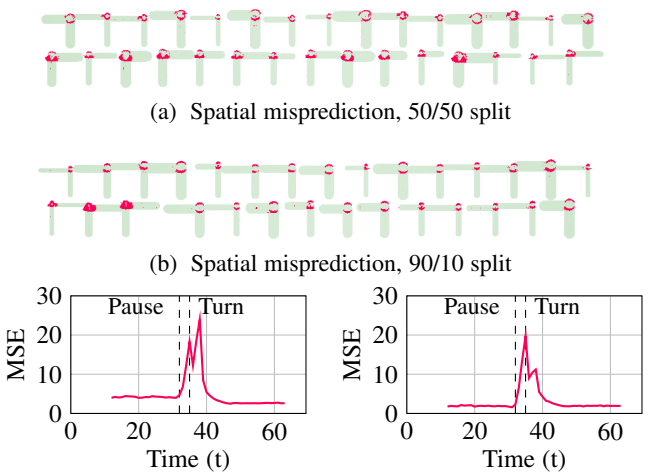
We can extract more information about the behavior irregularity by using not just one, but a set of models with varying capacity. Simpler models can only capture basic temporal relationships and tend to prioritize the most common scenarios, producing accurate predictions only in simpler spatiotemporal regions. More complicated models are able to represent many scenarios and produce fewer large errors, and thus focus on the most irregular behavior. This allows to roughly categorize different data regions in terms of their irregularity.

When doing prediction with multiple models we follow the approach described previously in Sec. 3.3. Only the extracted training data is reused by different models, while weight initialization, training and prediction are performed separately. Once we have made a prediction using each of the models, we compute the absolute differences to the original data, obtaining a misprediction volume for each model. We describe how these volumes can be used for visualization in Sec. 5 and Sec. 6.

4 Causes of Prediction Error

Before using the models and their prediction errors to support visualization, it is important to first understand their properties, which we discuss in detail in this section. There are several potential causes for errors of a local prediction model. We distinguish between three scenarios that can lead to high prediction errors, exemplify them via dedicated synthetic datasets, and demonstrate the corresponding results of our prediction-based approach:

- **Uncertainty (Sec. 4.1)** Behavior cannot be predicted completely from the input (due to stochastic processes or insufficient data).



(a) Spatial misprediction, 50/50 split

(b) Spatial misprediction, 90/10 split

(c) Temporal misp., 50/50 split (d) Temporal misp., 90/10 split

Fig. 2: Demonstration of the **uncertainty** scenario. The data has small circular objects with different radii moving upward with different, but uniform, velocities. After 32 timesteps and a pause of three timesteps, they randomly move left or right. We show the traces of the objects in pale green. Since neither the pause nor the turn are deducible from the local data, the outcome is uncertain, leading to prediction errors (in red). **a**: When two turn directions are evenly split, the model predicts a mixture of both turns, incurring large errors for most objects. **b**: When most of the objects turn left, the model predicts the left turn, incurring larger error on the right-turning objects. **c, d**: Average error in each timestep, both the pause and the turn events are visible. The error at the turn decreases for the 90/10 case, since the outcome is less uncertain.

- **Uniqueness (Sec. 4.2)** Behavior only rarely occurs (i.e., underrepresented in the training data).
- **Complexity (Sec. 4.3)** The model's capacity is insufficient to accurately fit numerous different behaviors.

4.1 Uncertainty

The uncertainty scenario arises when the target volume value is not completely determined by the model's input. This can happen due to hidden variables that are not included in the data, the patch size being too small to contain all the relevant information (in space or in time), or even due to innate stochasticity of the data-generating process. Conceptually, even an optimal predictor would have non-zero error in this scenario (Bayes error rate) [35].

To analyze this scenario, we constructed a dataset consisting of thirty small circular objects moving in space. The objects have randomized velocities and radii, but all move uniformly upward. After 32 timesteps, all the objects simultaneously stop moving, staying in place for three timesteps. Afterward, all of them continue the uniform motion but in different directions: half of the objects move to the left, and the other half moves to the right.

This dataset contains uncertainty, because it is impossible to predict whether an object is going to turn to the left or to the right using only local information. The pause in the objects' motion is introduced to signal the upcoming change of direction, in other words, the model "knows" when the turn is going to happen, but doesn't "know" whether it's a left or a right turn. By restricting the uncertainty to only the direction we are able to more clearly observe its effects.

We trained a small two-layer model using a prediction delay of three frames. Specifically, we use a "D64-D32" model throughout this section, with the model notation introduced in more detail later in Sec. 7. The spatial regions of high error and total error over time are presented in Fig. 2. The first important feature is the two peaks in the temporal view (Fig. 2c). The former peak corresponds to the pause in the objects' motion, since it cannot be expected from the local information alone (otherwise the model could "memorize" that it happens after 32 timesteps). The latter occurs when the objects continue to move left or right, and the model cannot predict the direction. In the spatial view (Fig. 2a) we show the traces of the objects (in pale green) and the errors (in red). We see that for many objects the model predicts some combination of the left and right motion, resulting in error blobs on the both sides of the turn. This shows that the model predicts the turn, since predicting a lack of motion would result in even larger errors, but it cannot predict the exact direction due to its uncertainty. Once the model observes the first of the frames after the turn, the uncertainty is no longer there, and the model's prediction becomes accurate again (thus there are no large errors after timestep 39).

For another experiment, we generated a similar dataset, where only three out of thirty objects turn right, and the rest turn left. We then trained the model with the same configuration as previously. Fig. 2d shows the temporal misprediction, where we can see that the second peak corresponding to the turn became significantly smaller. This can be explained by looking at the spatial misprediction in Fig. 2b: the three objects that turned right have large errors, because the model has been trained to predict the dominant left direction. The rest of the objects only have errors from the pause event (half-circles above and below, cf. Fig. 2a), thus resulting in an overall smaller error peak. Although the uncertainty is still present, the right turn scenario occurs less often in the data, and thus the average loss of predicting the left turn is significantly lower than the other alternatives (predicting a right turn or some mixture of both). Thus, the model always predicts the left turn which is the optimal prediction in this uncertain scenario (both in terms of the MSE loss and the Bayesian decision rule).

Note that we introduce the larger prediction delay to better illustrate the uncertainty scenario, but the results do not change qualitatively when using a prediction delay of one. The model still cannot anticipate the direction change and produces prediction errors. However, in this case they occur only during one frame and the error peak immediately follows the event.

4.2 Uniqueness

If a distinct pattern of local behavior is sufficiently unique, the model is likely to produce high errors predicting it. Since the behavior is underrepresented in the training data, it does not generate strong gradients, and thus the training process prioritizes more typical patterns. There is a tight connection to the generalization performance of the model and hence, regularization: accurately fitting an outlier implies overfitting the data, which becomes harder when using simpler models and/or stronger regularization.

To demonstrate the uniqueness scenario in practice, we constructed a simple dataset where twenty small objects of various shapes move uniformly upward with a constant velocity. The objects change their voxel values following a periodic function ("blinking"), with all but one object using a period of four

(a) Nineteen objects use a period of seven frames, the right-most object has a period of four.

(b) The left-most object uses a period of seven frames, the rest have period of four frames.

Fig. 3: Demonstration of the **uniqueness** scenario. The two datasets consist of objects with various shapes and sizes moving upward, changing their voxel values according to the same periodic function. We visualize the traces of objects (in pale green) and regions with high errors (in red). **a:** When one of the objects uses a different period than the rest, it is an outlier and causes high prediction errors. **b:** When all but one object use the former "outlier period", what used to be an inlier becomes an outlier instead.

frames. This single object is meant to be an outlier and has a period of seven frames.

The results obtained using the "D64-D32" model and a three-frame prediction delay are presented in Fig. 3a. There we visualize the traces of the moving objects (in pale green) and regions where the model produces high errors (in red). As we can see, the first nineteen objects (left-to-right) did not incur large prediction errors, while the last outlier object can be detected. In an inverse experiment, we used the four-frame period for a single object, and the period of seven frames for the following nineteen objects. The results for a model trained on this data are shown in Fig. 3b, where a symmetrically opposite outcome has occurred. The objects with the former "outlier period" of seven frames are now accurately predicted, while the four-frame object has incurred large errors.

This behavior might appear to be similar to the uncertainty scenario, however there is an important distinction. With uncertainty, two different outcomes follow identical (or very similar) local behavior. Thus, it is impossible for the model to "disentangle" the two outcomes in the input space, regardless of its capacity. In contrast, in the uniqueness scenario, the rare outcome is still completely determined by the prior local behavior.

4.3 Complexity

The complexity scenario refers to the situation where the model's capacity is insufficient to accurately fit the data. Conceptually, as the number of distinct local behavior patterns increases, the learned mapping is expected to produce a correct prediction over an increasing number of distinct regions of the input space. When the capacity of the model is too low, it cannot separate the different regions, leading to prediction errors. Although determining the capacity of a neural network is still a difficult problem [36], some understanding can be gained from statistical learning theory and the VC-dimension [37], which measures a classifier's capacity as its ability to separate arbitrary points in the input space.

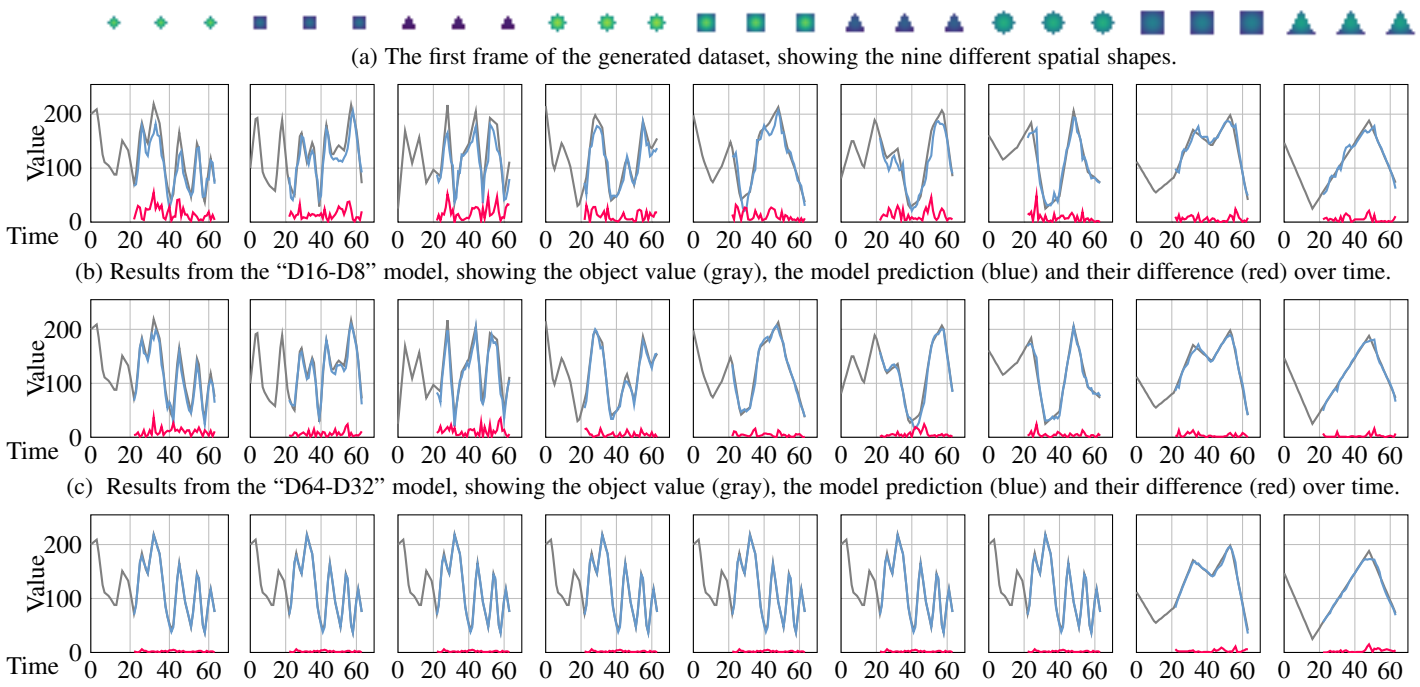
To illustrate the effects of complexity, we constructed a dataset consisting of 27 small objects of varying shapes and radii, which change their value following one of several predefined patterns. There are three possible shapes and three radii, resulting in nine unique object types. The first frame of the dataset is presented in Fig. 4a, demonstrating the shapes of the objects. For each of the nine shape-radius combinations we defined one unique temporal pattern. The patterns are polylines with a different number of segments, resulting in time series of varied complexity. Note that since each series corresponds to a unique spatial shape, there is no uncertainty in the data, and the future values are completely determined by the past. We trained a "D16-D8" model with a prediction delay of three frames on this data, using a long 20-frame patch size to minimize uncertainty of prediction. For convenience, we plot the prediction over time at the center point of nine objects, one plot for each unique shape (Fig. 4b). Even though the model was trained on the data, its capacity is insufficient to capture the many possible local behavior patterns. Thus, the model incurred significant errors (in red), with higher errors corresponding to the more complicated fast-varying temporal patterns.

In a complementing experiment, we trained a larger "D64-D32" model under the same configuration, with the results plotted in Fig. 4c. Although the data has not changed, the model was able to better predict the behavior, since its capacity is larger (overall MSE of 1.34 vs. 3.22 for the smaller model). Many more of the various patterns in the data are now captured well, thus highlighting a smaller subset of spatiotemporal events.

The complexity scenario is distinct from uniqueness, because all the behavior patterns occur with the same frequency. However, in practice they often interact, with both complexity and frequency of local patterns dictating what is predicted accurately by the model. We demonstrate this complexity-uniqueness interaction in Fig. 4d, where we modified the dataset, such that the most complicated time series is used more often than the simpler ones. Specifically, we replaced seven out of nine temporal patterns with the most complicated series from the previous experiment (on the left), leaving the two simpler patterns on the right unchanged. First of all, the overall MSE of the "D16-D8" model has decreased from 3.22 to 0.23 (cf. Fig. 4b), since there are fewer unique local patterns to capture. More importantly, the model has predicted the complicated behavior more accurately than the simpler one, because it is now common for the dataset. There lies a conceptual advantage of a machine learning model: it can still account for complex behavior if it is typical for a given dataset, without prior assumptions regarding the characteristics of this behavior.

5 Misprediction Visualization

Using our prediction-based approach (Sec. 3) we aim to capture the properties of the dataset in a domain-agnostic fashion, providing an overview of regions with irregular local behavior. For this, misprediction volumes obtained from multiple local models can be used to construct spatial and temporal views of irregularities in the data. Due to the stochastic nature of the model training process, the raw misprediction volumes are noisy, with the exact error magnitude varying among neighboring voxels. To suppress this noise and present a visually clear overview of each model's large-error regions, we apply spatial smoothing to each volume. This makes sure that large spatially-coherent prediction errors have a stronger effect on the results than sparse random deviations. For consistency, we use a kernel radius of five for all results presented in the paper. After the smoothing, we aggregate each misprediction volume separately into an aggregate volume using the maximum function. This way we avoid summing up smaller



(a) The first frame of the generated dataset, showing the nine different spatial shapes.

(b) Results from the "D16-D8" model, showing the object value (gray), the model prediction (blue) and their difference (red) over time.

(c) Results from the "D64-D32" model, showing the object value (gray), the model prediction (blue) and their difference (red) over time.

(d) Results from the "D16-D8" model on the modified dataset, where the most complicated time series was used for most of the objects. Showing the object value (gray), the model prediction (blue) and their difference (red) over time.

Fig. 4: Demonstration of the **complexity** scenario. **a:** The dataset consists of stationary objects with nine unique radius-shape combinations. Each object type "blinks" following a unique time series. **b:** Value at the center of nine objects (one for each unique behavior), shown together with the prediction of a simpler model (in blue) and the resulting error (in red). The more complicated fast-changing time series (on the left) result in larger and more frequent errors. The larger model in **c** produces smaller errors with fewer spikes, thus highlighting fewer spatiotemporal patterns. **d:** Interaction between the complexity and the uniqueness scenarios. When the most complicated behavior was made frequent in the dataset, the model captured it more accurately than the simpler one.

prediction errors, allowing us to distinguish between spatial regions where brief unpredictable behavior took place and regions where low errors consistently occurred throughout the dataset (see supplemental materials for a further discussion). Thus, we end up with multiple spatial volumes, each voxel of which represents the largest smoothed prediction error that a particular model has at the given spatial location.

Next, we assign a transfer function with a single distinctive color to each of the aggregated volumes, and perform multi-volume raycasting. We aggregate the samples along the ray front-to-back, compositing a sample from each of the volumes at each spatial location. This allows the user to distinguish where prediction errors occurred for each of the models. When compositing samples, we order the volumes according to their model's capacity, highest first. Although the transfer functions can be adjusted interactively, we often use single-peak transfer functions for iso-surface-like rendering of all but the first volume, which corresponds to the model with highest capacity. This way the user can always see the regions where even the most sophisticated model has failed, i.e. regions with the most unpredictable behavior. For context we also render the original spatiotemporal volume simply averaged over time. Here we typically use a relatively transparent ramp transfer function to keep the focus of the visualization on the prediction error volumes. We used this approach in Sec. 4 (e.g. Fig. 2a) and demonstrate the results in more detail in Sec. 8.1.

In addition to a spatial misprediction view of the dataset, we provide a temporal view. Its goal is to highlight the timesteps when the most unpredictable changes occurred and help to detect different temporal phases of the data. To this end, we aggregate the misprediction volumes separately for each model, for each timestep, eliminating the spatial dimension. The aggregation is done using the average function, which we found to be more appropriate for distinguishing temporal phases. As a result, we obtain a time series for each model, which we plot as a line graph. A previous example can be seen in Fig. 2d, while a detailed discussion of the results follows in Sec. 8.1.

The spatial and temporal views can be linked together via interaction for dataset exploration. Specifically, the user can interact with the temporal misprediction view by selecting a time range. When the range is selected, the spatial view is recomputed using only the timesteps that are part of the specified time range. This can be used to focus on periods of time with large prediction errors, providing a spatial view of the regions that couldn't be predicted well by the model. If only one timestep is selected, we show a single timestep of the original data augmented by regions of high prediction error without any aggregation. In this mode the user can access the lowest level of detail and search for explanation of the patterns occurring in the aggregated views.

6 Further Applications

In this section we discuss further applications of local prediction models in visualization, namely, adaptive timestep selection (Sec. 6.1) and a dissimilarity measure for ensemble data (Sec. 6.2).

6.1 Timestep selection

Spatiotemporal data often consists of many timesteps, all of which cannot be statically presented to the user, motivating temporal subsampling (e.g. [2], [3], [4]). However, a meaningful selection of timesteps requires either data-specific selection criteria or is based on quantification of timestep differences, whereas interesting process transitions may still be missed. Here, we aim to use the prediction error as a generic method to select timesteps of interest. Our goal is to automatically present the significant irregular temporal events to the user, which can act as an additional exploration approach, complementary to the misprediction views described in Sec. 5.

Our technique uses the misprediction volumes obtained from local models (Sec. 3.2) to detect timesteps that contain irregular temporal events. As usual, we begin by training a local prediction model and measuring the prediction error on the original data. As described in Sec. 5, we average the misprediction volume spatially, constructing a time series for the model. The resulting time series describes the mean prediction error that occurred at a certain timestep (e.g. Fig. 2d).

For timestep selection, we now consider the maxima of the time series, since they mark irregular temporal events and transition points in the data. To remove smaller deviations and help focus on the most significant events, we perform bilateral filtering of the time series, using a Gaussian kernel for both the time and value dimensions. The overall effect of bilateral filtering is smoothing over time that avoids smoothing over elements with large differences in value, thus preserving the significant error peaks that we are interested in. Then, we extract the local maxima points of the series and remove consecutive points that are closer than a threshold. This helps to avoid choosing peaks that are only a few timesteps away, which can sometimes happen for events that occur over several timesteps and cause some lingering fluctuation of the prediction error. Generally, we set the threshold to only a few frames, to avoid missing separate consecutive events. After the maxima are detected, we pick the corresponding timesteps as ones containing significant temporal events.

6.2 Ensemble data analysis

Ensemble data is a new challenging direction in visualization research [38]. Many approaches have been proposed ([39], [40], [41], [42]), however, very specialized methods are often required to address the problem of such complexity. Sedlmair et al. [43] present an overview of recent ensemble visualization approaches. Some of the common tasks in ensemble analysis are partitioning of the output space, outlier detection and sensitivity analysis, all of which require a measure of distance or similarity. In a simple case of scalar outputs the difference between outcomes could be easily quantified. But if the simulation produces a whole spatiotemporal volume, straightforward distance measures such as voxel-to-voxel difference do not produce a meaningful result in many cases (e.g. [4], [17]). For example, a phase shift between two members would produce a large quantitative difference, while qualitatively they are very similar. Therefore, we need higher-level similarity measures and that often requires specialized techniques and domain- or even data-specific knowledge.

We propose a domain-agnostic dissimilarity measure for ensemble members that could be used to augment existing specialized techniques. The measure is an extension of our local prediction approach. The main idea is to perform *cross-prediction* across the ensemble members: training a local prediction model on one member and then applying it on another, measuring the overall prediction error. This effectively estimates the difference between the behavior captured by the model from one member and the actual behavior of another member. Conceptually, the measured error can also be thought of as the generalization error of the model: by observing how well the model performs on unobserved data, we are estimating how different the data is from the training dataset.

Specifically, we train a separate local prediction model (Sec. 3.2) on each member of the ensemble. Then, using each model, we perform prediction for each member (including the one that the model was trained on) and compute the mean squared prediction error across the whole spatiotemporal domain. Thus, we end up with a square matrix of cross-prediction errors E, where each cell $E(i,j)$ specifies the error resulting from training on the i-th member and predicting for the j-th member. We further normalize the cross-prediction error by the error measured when training on the same data. This addresses the fact that some members can be significantly harder to accurately predict than others due to a higher amount of irregular behavior, regardless of which data the model was trained on (Sec. 4). This could lead to cross-prediction errors being large for a similar pair of complex simulations and low for a similar pair of simple simulations, although qualitatively the dissimilarity within both pairs should be the same. Therefore, we use the *relative cross-prediction error* E_r:

$$E_r(i,j) = E(i,j)/E(j,j). \tag{4}$$

Naturally, the cross-prediction error is not symmetric. This is not only a result of model training being a stochastic process, but also an important insight into the meaning of the error. A large cross-prediction error means that local behavior common to the predicted member didn't occur in the training member (or occurred too rarely). This implies, that training on a simulation with diverse local behavior may produce a low cross-prediction when predicting for another member. The predicted member on average may be very different, but it's local behavior is a subset of what the model has captured during training and can reproduce during prediction. Thus, to construct a dissimilarity measure from the cross-prediction error we consider the error in both directions:

$$\overleftrightarrow{E_r}(i,j) = \overleftrightarrow{E_r}(j,i) = \sqrt{E_r^2(i,j) + E_r^2(j,i)}. \tag{5}$$

This means that the dissimilarity between a pair of datasets is only low if both cross-prediction errors are low.

Having a dissimilarity measure for the ensemble, several visualization techniques could be applied. We chose to visualize the ensemble using a dissimilarity matrix encoded as a heatmap. This allows us to better evaluate our approach, since we directly visualize the values of our measure for different member pairs. To improve the visualization we also sort the ensemble members using hierarchical clustering. Specifically, we perform agglomerative hierarchical clustering with centroids, i.e. when merging clusters, distance between clusters is defined as distance between the cluster centroids. The resulting tree is then sorted, such that the distances between adjacent leafs are minimized [44]. In other words, the ensemble members in the visualization are arranged such that the dissimilarity between neighbors is small.

7 Implementation

All models in this work are feed-forward neural networks with two types of layers: convolutional and densely-connected. Convolutional layers act on spatial dimensions using filters of size $3 \times 3 \times 3$.

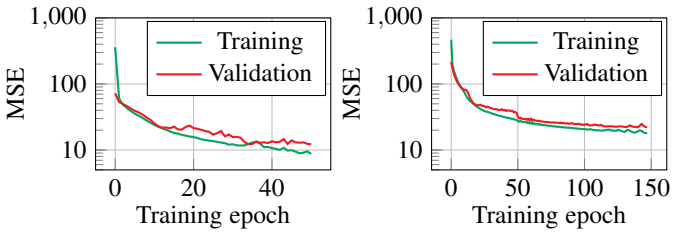
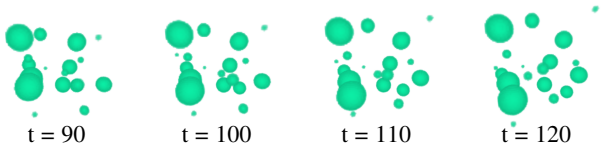
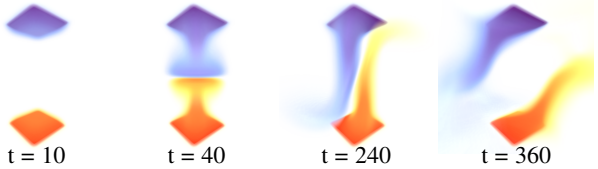


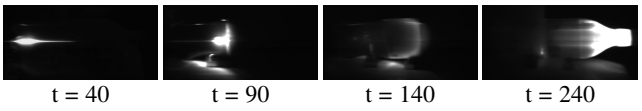
Fig. 5: Training history for the models "C64-C64-D256-D256" (left) and "D256" (right) on the droplet dataset. We perform early stopping to avoid overfitting, terminating the training when the loss on the validation dataset has stopped improving.



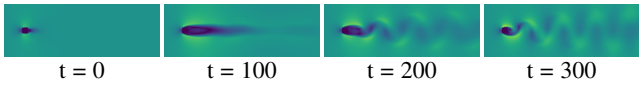
spheres: synthetic data containing moving spheres with collisions.



hotroom: air temperature simulation.



bottle: images of a laser pulse propagating through a bottle [45].



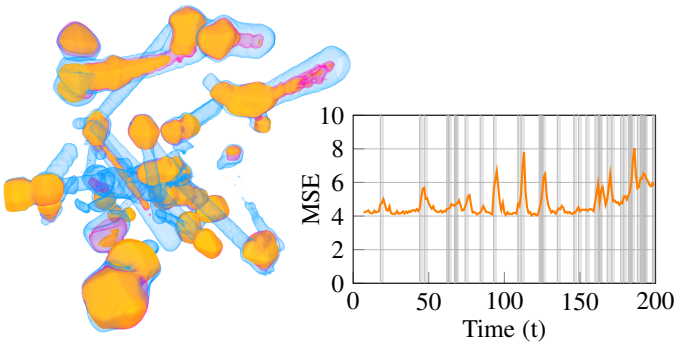
vortex street: CFD simulation of flow around a cylinder.

Fig. 6: Some of the datasets used in our evaluation.

For 2D datasets, filters of size 3×3 are used. In model descriptions we refer to convolutional layers as "CX", where X specifies the number of feature maps. We refer to dense layers as "DX", where X is the number of neurons in the layer. We utilize ReLU activation for both layer types. When using a combination of convolutional and densely-connected layers, we insert a non-parametric flattening layer in-between, which reshapes the input 4D array into a flat array appropriate for dense layers. Additionally, all models have a final dense layer with a single neuron and no activation, which acts as a linear output unit for regression.

For instance, model "C64-C64-D256-D256" has the following layer sequence: "Convolution(64, 3^3), Activation(ReLU), Convolution(64, 3^3), Activation(ReLU), Flat(), Dense(256), Activation(ReLU), Dense(256), Activation(ReLU), Dense(1)". A special case is the model we refer to as "D1", which in fact consists only of a flattening layer and a single neuron with no activation function. Thus the model learns a linear function of the supplied inputs.

We normalize the training data to mean zero and standard deviation of one, but otherwise do not perform any pre-processing assuming the data to be clean. Should the dataset contain a significant number of broken or missing cells, we could provide a boolean mask as an additional input feature to the model.



(a) (b)

Fig. 7: **a:** Spatial misprediction on the synthetic dataset with moving spheres. The most complex model (orange) detects sphere collisions, while simpler models (magenta, blue) also highlight some of the faster and irregular motion. **b:** Comparison of detected temporal events to the ground truth. **Orange:** Mean prediction error of the model in each timestep. **Vertical lines:** Timesteps when events happen. Light-gray stripes show the expected error delay (two frames). Whenever an event occurs, we observe a corresponding spike in the prediction error.

We train all our models using the "Adam" [46] variation of the stochastic gradient descent algorithm, with learning rate 0.001 and batch size 1024. Following standard practices, the training is performed in epochs, where in each epoch the model observes the whole training dataset once. We perform holdout validation, splitting off 20% of our data into a validation dataset to monitor the generalization performance. Furthermore, we use the validation data to perform early stopping as a form of regularization [36], although due to the relatively small capacity of our models and large amounts of data we do not experience severe overfitting showing similar training and test losses (Fig. 5). We monitor the validation loss and stop the training process when no improvement has been observed for 25 epochs and take the model checkpoint from the best epoch as our trained model.

To perform prediction on a dataset we need to evaluate the trained model on every spatiotemporal patch. Even for medium-sized volumes the amount of input data may reach terabytes. Because of this, our prediction implementation operates out-of-core. We extract batches of spatiotemporal patches from the original data stored on disk, compute the prediction on the GPU and write the results back to disk. An additional optimization technique that we use is caching of the prediction result for spatiotemporal patches that contain nothing else but empty space (in the input and in the target). As a result, we obtain identical prediction results, but can reduce the GPU execution time for some of the datasets.

8 Results

We now present and discuss the results of our prediction-based visualization approach, including the spatial and temporal misprediction views (Sec. 8.1), adaptive timestep selection (Sec. 8.2), and ensemble dissimilarity analysis (Sec. 8.3). We also study the effects of several important parameters (Sec. 8.4), demonstrate our parameter selection method (Sec. 8.5) and extend our implementation to multivariate volumes (Sec. 8.6) The datasets used in this section are shown in Fig. 6 and videos with the rendered results are provided as supplemental material.

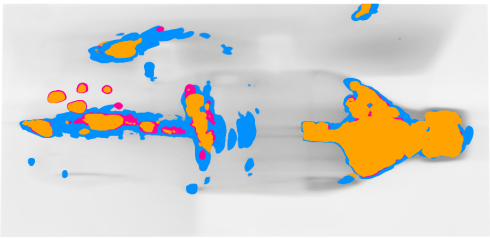


Fig. 8: Spatial misprediction view of the "bottle" dataset, with original data for context in gray. The complex model (in orange) highlights only the transition regions, while the other also show the initial pulse trajectory (on the left). The diffuse phase is completely filtered out (in the middle).

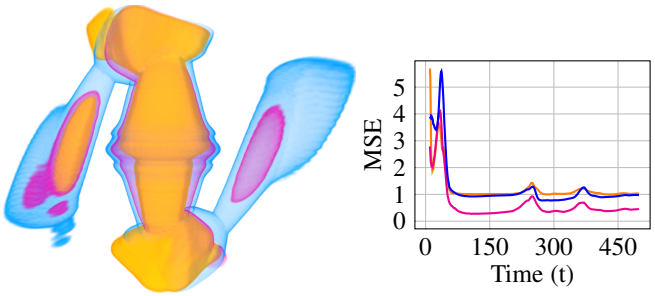


Fig. 9: Spatial and temporal misprediction views of the "hotroom". Three significant events can be identified in otherwise smooth data: the initial front collision ($t = 21$), collision with the plates ($t = 231$) and the formation of new streams ($t = 350$).

8.1 Spatial and temporal misprediction views

First we present our visualization approach to the analysis of time-dependent volume datasets (Sec. 5). We use the following three model sizes throughout this section: "C64-C64-D256-D256", "D256" and "D1". For brevity and uniformity, we refer to them as *"Model A"*, *"Model B"* and *"Model C"* respectively. We also introduce a short patch size notation: "l_t x l_s^k", where l_t and l_s are temporal and spatial extents (Sec. 3.3) and k is the number of spatial dimensions, e.g. $7x5^3$ is a patch of size 7x5x5x5.

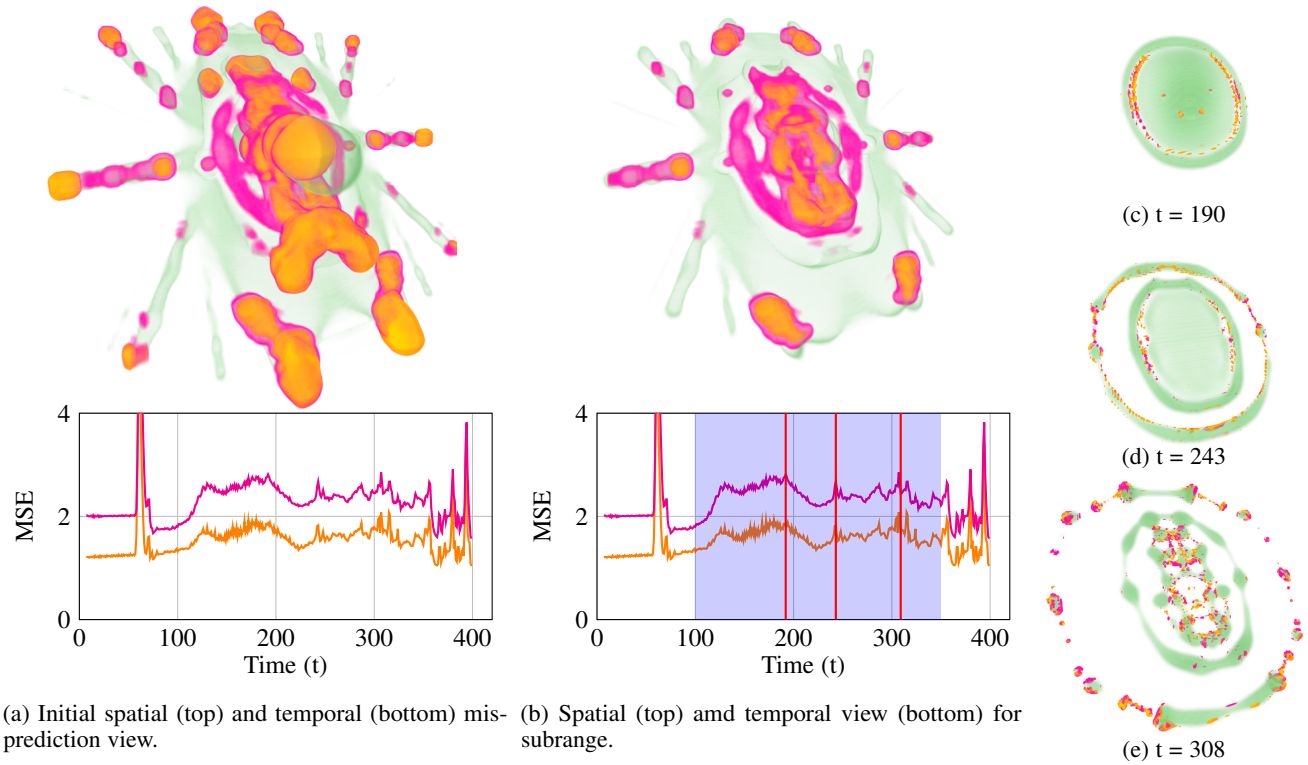
Synthetic dataset (Fig. 7). We begin with a synthetic dataset for which we can provide the ground truth describing all the events that are happening in the data. The generated dataset contains smooth spheres traveling along straight trajectories. Spheres fully-elastically react to collisions with each other and with the boundaries of the dataset (Fig. 6). Some of the spheres shrink or grow as they move, and may abruptly change their trajectory. Overall, the data contains simple behavior (linear motion, size change) as well as more interesting and hard-to-predict events (collisions and trajectory changes).

We applied models A, B and C to the data and present the resulting spatial misprediction view in Fig. 7a. What we can see is that the model with the highest capacity (orange) produces large errors only in smaller regions. These regions correspond to sphere collisions (appearing as two adjacent objects) and significant trajectory changes. The model was able to adequately capture the behavior associated with motion, and thus isolated the collisions and sharp trajectory changes. A simpler model (B, in magenta) failed in some additional regions, corresponding to trajectories of smaller fast-moving objects. Finally, the simplest model (C, in blue) displays significant prediction errors for many faster-moving objects, resulting in long tube-shaped regions representing the linear trajectories of the spheres. Another interesting property to observe is that the high prediction error regions are often subsets of each other: the simple model fails where complex model fails, but also in some additional locations. This is expected behavior, considering the fact that all model architectures are (1) qualitatively similar (in terms of their components) differing mostly quantitatively, and (2) trained on the same data with the same loss function. Data points that produce large gradients during training result in highest "priority" for all models. The simpler models do not have a high capacity for capturing different scenarios, so after fitting the most "important" data points, they cannot fit additional scenarios without worsening their overall performance.

In Fig. 7b we present the temporal misprediction view for our synthetic dataset obtained using the model A. To compare our results to the ground truth, we plot the model's error (orange) and mark the timesteps where an event has occurred with a gray vertical line. The figure shows that when no events happen the model maintains a steady prediction error. However, when an event takes place, we can see a corresponding spike in the graph. Another interesting observation is that the peak error occurs several frames later than the event itself. This is due to the prediction delay (Sec. 3.3), which in this case was three frames ($d = 3$). To better illustrate this effect we have also plotted the expected error delay with a wider stripe. In the frame when a collision happens, the objects change their movement direction, but still haven't traveled too far apart from their original trajectory. In the next $d-1$ frames, the model still hasn't observed the collision (because of the prediction delay), so it continues to predict movement along the original trajectory, while the object continues to move away, increasing the error. Once the model has seen the collision, it "corrects" the prediction, and the error drops to its normal level. The drop is also not immediate, since the model initially sees only one frame of the new trajectory, and typically needs several timesteps to accurately extrapolate the object's future position.

Bottle (Fig. 8). In Fig. 8 we show our spatial misprediction view of the bottle dataset. The data contains femto-photography images of a laser pulse propagating through a plastic bottle [45] (Fig. 6). We used models A (in orange), B (in magenta) and C (in blue) to perform the prediction. As a result we obtain a visualization highlighting regions of transitional behavior (Fig. 8). Roughly speaking, the pulse moves from left to right, exhibiting different phases: appearance of the initial pulse, transition into the diffuse phase behind the bottle label, appearance of inter-reflections near the neck of the bottle. Importantly, the complex model (in orange) highlights only the transitional regions. Both the propagation of the diffused pulse (in the center), and the propagation of the inital pulse (on the left) are not exhibiting irregular behavior and are captured by the model. The simpler models have a harder time predicting it, resulting in a difference between the misprediction regions. Similarly, the areas in the upper part of the dataset correspond to the appearance of the reflection of the pulse (detected by all models) and its propagation (shown by the simplest model in blue).

Hotroom (Fig. 9) We use the "hotroom" dataset (Fig. 6) to show our results on spatially and temporally smooth data. It depicts a simulation of air temperature in a room with a hot and a cold plate on the bottom and the top, respectively. This produces a dense smoothly-changing temperature field (we cut off neutral temperatures in the transfer function to highlight the cold and



(a) Initial spatial (top) and temporal (bottom) misprediction view.

(b) Spatial (top) amd temporal view (bottom) for subrange.

Fig. 10: Interactive analysis of the "droplet" dataset. **a**: The models (orange and magenta in spatial view) highlight the initial droplets, the expanding ring, a few larger droplet separations and several oscillating droplets. Two largest spikes in the temporal view correspond to simulation timestep changes. **b**: After selecting a subrange, we no longer see the inital droplet collision and can better focus on the locations of droplet separation. **c, d, e**: Single-timestep views showing exact locations of the prediction error (see vertical lines in **b**).

the hot air streams in Fig. 6). We use models *A*, *B* and *C* to construct spatial and temporal views presented in Fig. 9. The spatial misprediction regions correspond to three major events that are also apparent in the temporal view: (1) the initial collision of cold and hot streams produces (pillar-like structure in the middle), (2) the collision of the streams with the plates of the opposite temperature (asymmetric regions top and bottom), and (3) the formation of two new air streams, after the initial fronts have disappeared (side regions).

Droplet (Fig. 10). To demonstrate the interaction between the temporal and spatial misprediction views, we have applied our method to the droplet dataset. The dataset comes from a two-phase flow simulation of two colliding droplets. After the initial collision, the droplets form an expanding disk of fluid that consequently splits into many secondary droplets. Drop collisions are highly relevant in many technical applications, e.g., concerning fuel injection and fire suppression. The data contains both temporal and spatial events, providing an interesting test case for our approach. For our analysis, we consulted with the domain scientists from the field of aerospace thermodynamics who conducted the simulation.

Fig. 10a presents the temporal misprediction view of the data. Although a lot of smaller events happen in the data, we can distinguish the initial two-droplet phase with steady error (frames 0-70), disk expansion phase with increasing error (frames 70-130), and a long tail of smaller spikes with an overall decrease in error, as many droplets separate (causing spikes) and continue to travel predictably (reducing the overall error). Of special interest to us were the two biggest spikes at frames 62 and 394. The model's prediction contains massive errors that couldn't be explained by the data. After consulting the domain scientist we discovered that the simulation has irregular timestep sizes, and the two spikes in model's error align with two abrupt changes of the simulation timestep, effectively highlighting an irregular temporal event. In Fig. 10a we also show the spatial misprediction view of the data, obtained using the model *A* (in orange) and the model *B* (in magenta). We also aggregate and render the original data in faint green for context. To filter out the initial droplet phase and the discovered simulation timestep changes, we select a temporal subset of our misprediction data (Fig. 10b). Inspecting the corresponding spatial view, we see several locations of irregular behavior: the central regions, where many secondary droplets have separated, the large ring corresponding to the ring separation, and four smaller regions in the corners, where several large droplets have formed and split off. We also see abrupt traces of several large droplets flying off to the left and to the right, caused by their irregular oscillating motion. To investigate further, we focus on a few smaller temporal events, inspecting three individual timesteps (red lines in Fig. 10b). The resulting spatial views (Fig. 10c, 10d, 10e) show where the high errors occurred. In Fig. 10c and Fig. 10d we see the source of the ring observed in the aggregated view: this is the location where rings of fluid have separated from the initial expanding disk. Fig. 10e highlights the secondary droplets that split off in the center, as well as the four large droplets starting to form and separate, producing the corner regions in the aggregated view. Interestingly, we found an unexpected feature in the data: there are four small regions of error in the center of Fig. 10c, which we could not explain. After an investigation together with the domain scientist it turned out, that the fluid disk contains small bubbles

Fig. 11: Timesteps of the droplet dataset that were automatically selected by finding maxima of smoothed model error (cf. Fig. 12). High error locations are rendered in red. Several sparse important events are captured in the beginning: the initial collision, two disk separations, outer disk break-up. The rest of the timesteps correspond to a more chaotic mass separation phase of the data.

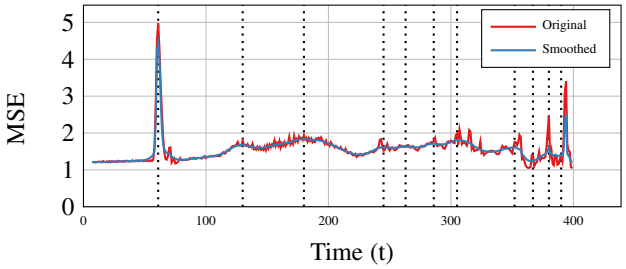


Fig. 12: Smoothed prediction error on the "droplet" dataset. By finding maxima of the smoothed prediction error we are able to find key irregular events in the data.

of air caught during the collision. Its existence was previously unknown and undetected using domain scientists' current tools: volume and vector visualization in ParaView and histograms of droplet properties extracted via scripts (mass, surface, etc.). The discovery of the irregular timesteps (believed to be an issue with the pressure solver), as well as of the bubbles (known neither to us nor the domain scientist) informally demonstrates the potential of our domain-agnostic approach to detect unexpected data features. Overall, navigation of the misprediction volumes allows us to get a quick impression of the most complex temporal and spatial regions, and then to use this information to "drill down" into the data, studying individual spatiotemporal events.

8.2 Timestep selection

Next, we exemplify the results of our timestep selection approach (Sec. 6.1) by means of the droplet collision dataset. We used the "C64-C64-D256-D256" model to construct a misprediction volume. The respective temporal misprediction graph is plotted in Fig. 12. The time series was smoothed using bilateral filtering, using a Gaussian kernel with STD of 5.0 for time and 1.0 for value. The selected timesteps (corresponding to the error maxima) are marked with vertical lines, and their renderings are presented in Fig. 11). The first few selected timesteps represent several important events: droplet collision, first ring separation, second ring separation, and the break-up of the ring into multiple droplets. The later timesteps correspond to several smaller separations and oscillating droplets. We can see that our method chose timesteps around the time of large key events and more densely sampled the temporal phase with many smaller interactions. This technique could be used to automatically select most significant irregular events, if manual navigation of the misprediction views is not desired.

8.3 Ensemble data analysis

To evaluate our prediction-based ensemble dissimilarity measure, we have applied it to a CFD ensemble (similar to the "vortex street" dataset). The ensemble has three parameters: Reynolds number,

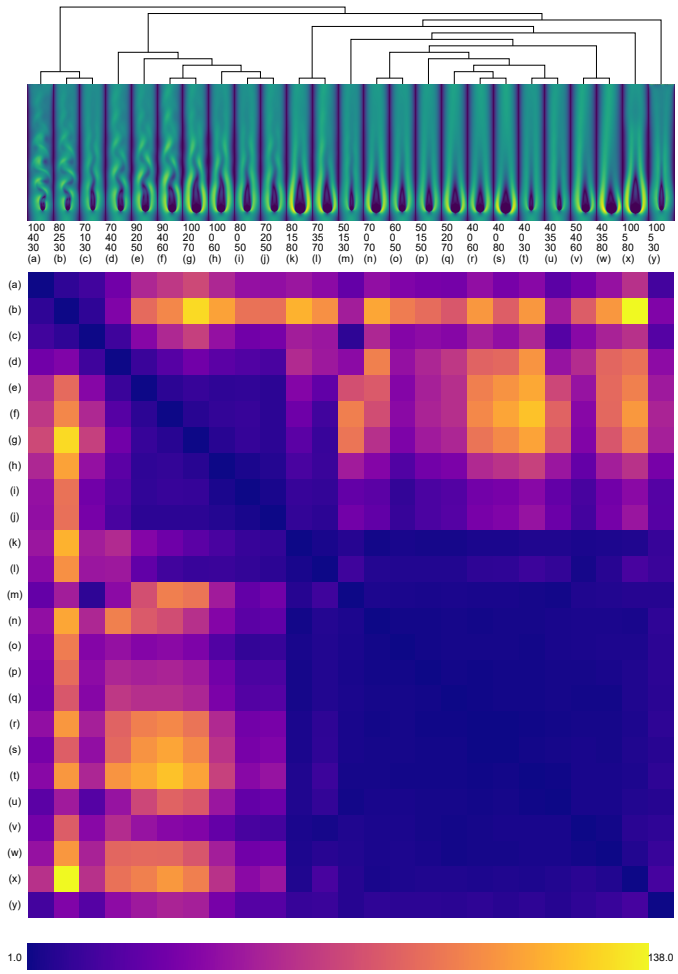


Fig. 13: Our dissimilarity measure for a flow simulation ensemble. The top row shows one timestep from each simulation. Groups of members that exhibit similar behavior (e.g. turbulent) have low dissimilarity (blue) to each other, but high dissimilarity (yellow) towards other members.

obstacle offset and obstacle radius. To obtain a diverse set of simulations we have randomly sampled the whole parameter space, obtaining 25 simulations.

For computing our dissimilarity measure (Sec. 6.2) we trained an instance of the "C128-D256" model for each member, and performed a full set of cross-predictions (625 predictions in total). The results are presented in Fig. 13. In the header we visualize a single timestep of each ensemble member and specify its three simulation parameter values. The same timestep has been used for all members, aiming to give an impression of the whole ensemble.

The first observation is that the dissimilarity metric led to ensemble members being roughly sorted according to how turbulent

they are. The most turbulent members were put on the left, with members slowly becoming more laminar as we move to the right. Looking at the dendrogram in the header, three high-level clusters can be distinguished: strongly-turbulent (on the left), turbulent (middle) and mostly-laminar (right).

When inspecting the heatmap, several groups of simulations can be spotted. The most prominent group of members is (d - j), which has high dissimilarity to many other members outside of the group. This is due to their highly turbulent behavior that doesn't appear in other simulations. Importantly, the dissimilarity inside this group is low, showing that our measure captures the differences in local behavior, rather than just differences in value (direct difference between two turbulent members would still be high). Another significant group of simulations is (m - x), which represents members with mostly laminar behavior. Again, they have low dissimilarity in-between, and higher values to other members, especially towards the highly turbulent ones. Within the groups we can look at a few members that were added to the cluster later (because of their slightly higher dissimilarity to the rest), e.g. (v) and (w) that are significantly asymmetric.

Simulations (k, l) also form a low-dissimilarity pair and represent a less distinct case of slower turbulent results. We observe a medium dissimilarity to the most turbulent members, as well as to the mostly laminar members. Presence of some turbulence helps the models trained on them to perform better on the turbulent members than the laminar-trained models. Similarly, member (o) is also interesting in that it exhibits an in-between case: although it belongs to the group of laminar members, it has lower dissimilarity to the turbulent ones than the rest. During most of the simulation time it exhibits laminar flow, but turns turbulent towards the end. For this reason, the model trained on simulation (o) was more successful in predicting the turbulent ensemble members, akin to (k) and (l). Another special case is simulation (b) which is the most turbulent member of the ensemble with very distinct local behavior that leads to large prediction errors for most models. Simulation (c) is similar to it in terms of turbulence, however, it has a longer laminar setup phase, leading to lower dissimilarity values overall (laminar models can still predict the first part correctly). Finally, interesting outliers are members (a) and (y), which were sorted away to the left and right sides by the hierarchical clustering algorithm. Initially, they appear as a strongly-turbulent member and a laminar member respectively. However, closer inspection showed that the simulation files were corrupted, resulting in "flickering" that hasn't been observed in the ensemble before, but was identified via our visualization. These artifacts caused increased errors when predicting on the data, even for the model trained on it, thus resulting in medium normalized cross-prediction error for most other ensemble members. Overall, we found that our dissimilarity metric produces results that correspond to what is intuitively and qualitatively expected from comparing pairs of ensemble members, which allows us to identify prominent features of the ensemble.

8.4 Parameter study

Next, we analyze the effect of our parameters. We look at the prediction delay in Sec. 8.4.1 and at the patch size in Sec. 8.4.2.

8.4.1 Prediction delay

Prediction delay (Sec. 3.3) is the distance in time between the last timestep provided to the model and the predicted timestep. To demonstrate its effects, we perform prediction with ten "C64-C64-D256-D256" models, each using a different delay on the "vortex street" dataset, which has very fine temporal resolution. The temporal misprediction view for each delay is presented in Fig. 14a. As we can see in the cases of one- and two-frame delays, low prediction delay has a very clear symptom: model prediction error exhibits almost no variation. Since the prediction task is too easy, the model is able to predict the whole dataset equally well. When we increase the delay, we can observe an overall increase in the error, as well as more pronounced differences between temporal phases. Crucially, despite the quantitative variation between the different series, all models trained with the prediction delay of more than two frames highlight similar temporal events: growth of the trail and of the error until timestep 180, transition to more turbulent behavior until timestep 400, and eventually the onset of fully-periodic behavior indicated by decreasing model error.

8.4.2 Patch size

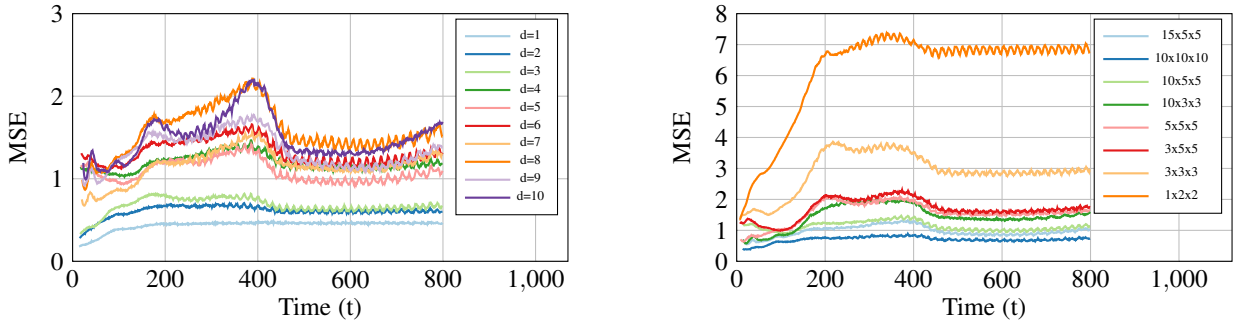
Patch size is an important parameter for a local prediction approach, since it defines what is local, i.e. what information is available to the model. For consistency, we use the "vortex street" dataset and perform prediction with eight "C64-C64-D256-D256" models. The results are presented in Fig. 14b. An important property of prediction for this fairly regular dataset is whether we can distinguish the different temporal phases from the model's misprediction. The "contrast" between the setup and the periodic phases varies with different patch sizes, but overall most models produce reasonable results. However, there are two exceptions. First, a very restricted patch size of 1x2x2 produces a constantly growing error graph, showing that even high-capacity models cannot produce an accurate prediction given insufficient information. Second, the very large 10x10x10 patch produces low error compared to other models, but the misprediction graph of this model alone shows much smaller differences between the temporal phases. A large model with a very wide view can predict well even the complex setup phase.

8.5 Parameter selection

After discussing our two most important parameters and how they impact the method (Sec. 8.4), we demonstrate our semi-automatic approach to their selection.

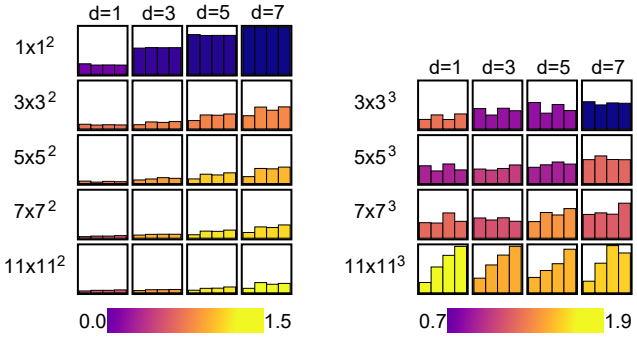
The patch size and the prediction delay parameters control the difficulty of the prediction problem: large delay (further ahead prediction) and small patch size (less input) make the prediction more difficult, while small delay and large patch size make it easier. Since our technique differentiates spatiotemporal regions based on prediction errors, it functions best between the extremes of impossible and trivial prediction, when some regions are hard to predict and some are not. A key observation is that as we approach these extremes, models of different capacity tend to produce similar errors. When the prediction is too difficult, all the models display similarly large errors, because additional model capacity does not help when information is missing. And when the prediction is too easy, all the models display similarly low errors, since even the simplest models can predict slow smooth processes. Thus, to perform parameter selection we compare the models' prediction errors and maximize their diversity, favoring settings that lead to larger differences between simple and complex models.

Specifically, we run our method using multiple models and compute misprediction volumes as described in Sec. 3.3. Then, for each spatiotemporal position we compute the diversity as the difference between the smallest and largest errors for this point among the different models. Then, we average this value over the



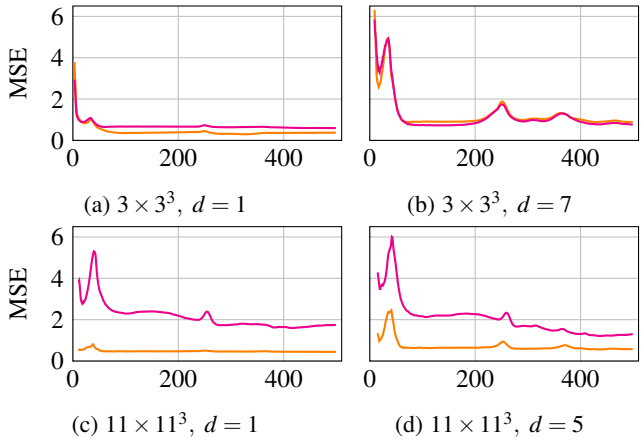
(a) Prediction delay (b) Patch size

Fig. 14: Parameter study on the "vortex street" dataset, showing temporal misprediction plots. **a:** Models with a delay below three frames do not detect any events because it is easy to predict a few frames ahead.**b:** Most of the models successfully highlight the complexity of the setup phase (frames 100-400). However, the models with very small (in orange) or very large (in blue) patch sizes do not clearly distinguish it.



(a) "vortex street" (b) "hotroom"

Fig. 15: Our parameter space visualization for two datasets. We show a grid of bar charts for different values of patch size (rows) and prediction delay (columns). Each bar chart encodes the mean errors of the four models (sorted complex to simple, left to right). Additionally we compute an error diversity criterion which we use to color each cell. Higher delays with larger patch sizes lead to higher error diversity, which indicates a prediction problem of appropriate difficulty.



(a) 3×3^3, $d = 1$ (b) 3×3^3, $d = 7$

(c) 11×11^3, $d = 1$ (d) 11×11^3, $d = 5$

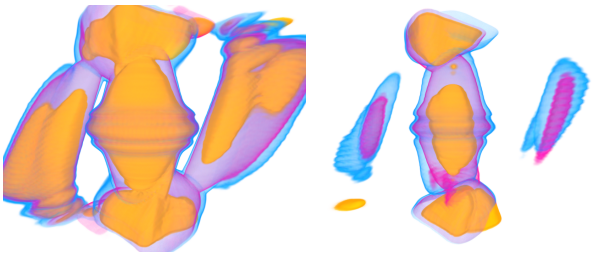
Fig. 16: Temporal misprediction for four different parameter configurations on the "hotroom" dataset (cf. Fig. 15). With low prediction delay (**a, c**) prediction is too easy and leads to models showing few errors, missing an event around frame 370. With small patch size (**a, b**) there is little difference between simple and complex models. Larger patch size and delay (**d**) allow for detection of all three events and distinction of temporal differences between the models.

whole dataset and divide it by the average prediction error of all the models, effectively computing the relative average prediction error range as our selection criterion. The normalization allows for better comparison across different configurations. Overall, larger error ranges indicate better parameter settings.

Using this information we construct a parameter space visualization. We render a grid, where each row corresponds to a patch size value and a column to a prediction delay value, and show a bar chart in each cell. The bar chart displays the average error of each model (ordered complex to simple, left-to-right), allowing the user to see a summary of models' performance. Additionally, we color each bar chart (grid cell) according to our error range criterion, showing how appropriate each configuration setting is in terms of local error diversity.

In Fig. 15a we show results on the "vortex street" dataset, using models "C64-C64-D256-D256", "D256", "D64-D32" and "D32-D16" (left to right). Here we also include the trivial patch size setting of $1x1^2$. First, as to be expected when considering prediction difficulty, larger patch size and lower prediction delays lead to lower overall errors. Importantly, when the patch size is too small ($1x1^2$) or the delay is too low ($d = 1$) the models show very little diversity, either failing or succeeding together. But when the patch size is large enough, larger delays lead to more diversity, since more complex models can still predict well enough, while the simpler models fail. Here specifically we find that patch sizes above $5x5^2$ and delays above 5 lead to best results.

Next, we performed experiments on the "hotroom" dataset, using the same four models as before. Since 3D datasets are much larger and generate a lot of input data, we introduced stronger patch undersampling ($p_u = 10^{-4}$), which leads to somewhat noisier results, but still allows trends to be studied. Thus, we can perform a parameter study faster, and then if needed run the core method with more data using the chosen parameter values. We demonstrate the results in Fig. 15b. Overall, we see a similar pattern: small patch sizes lead to uniformly large errors, while larger delays improve error diversity, especially for larger patch sizes. In this case, all the settings with $11x11^3$ patch size lead to larger error ranges, this is explained by fact that very simple non-convolutional models



(a) $3 \times 3^3,\ d = 7$ (b) $7 \times 7^3,\ d = 5$

Fig. 17: Spatial misprediction views for configurations with low (**a**) and high (**b**) error diversity on "hotroom". When the prediction problem is hard and diversity is low, different models produce many large errors and highlight similar regions, however when the problem has suitable difficulty we can distinguish regions of different irregularity.

tend to perform worse on very large 3D patches. For the "hotroom" we find patch sizes of at least $7x7^3$ and the delay of 5 as the most appropriate. In general, we recommend picking the largest patch size that is computationally feasible and the delay value that maximizes model error diversity.

We also investigated temporal misprediction graphs for four different settings using the most complex and the simplest models (Fig. 16). As we can see, runs with small delay (Fig. 16a, 16c) tend to have very smooth low-error curves and miss the stream emergence event around frame 370. The run in Fig. 16b (due to its larger delay) highlights all three events, but both models provide identical information. The configuration with larger patch size and large delay (Fig. 16d) further improves this result, with the complex model clearly distinguishing the three significant events, and the simpler model also showing a decreasing error trend as the air becomes more diffused. In Fig. 17 we also show the spatial misprediction views for two configurations: one with low and one with high error range criterion values. We observe that when the prediction problem is too hard (Fig. 17a), similarly to the temporal view (Fig. 16b), all models fail more often and in similar regions. But when we increase the patch size, and thereby error diversity, we can better separate different irregular regions, e.g. regions on the sides corresponding to later air stream formation are not showing errors from the most complex model (in orange).

Crucially, as discussed above, most parameter configurations yield similar results, with particularly poor settings being easy to avoid using either our selection method or minor prior knowledge of the data. Furthermore, we found that the parameter space visualization provides further insight into the data and can be used to complement our core technique, e.g. we see that for the "hotroom" prediction delay has a much smaller impact on model error compared to "vortex street", which points to a much smoother and predictable nature of the data.

8.6 Multi-field volumes

So far we have concerned ourselves with prediction on scalar volumes. However, data coming from scientific simulations is typically multifield, containing velocity vectors, pressure, temperature and other relevant properties that may be essential for the prediction task. In this section we supply additional fields as input to the model and investigate how this affects the prediction, and thereby the detected spatiotemporal regions.

Dataset	Pred. delay	Model	MSE
cylinder multi-field	1	C64-C64-D256-D256	5.21
cylinder scalar	1	C64-C64-D256-D256	15.53
cylinder multi-field	1	D64	10.93
cylinder scalar	1	D64	14.03
cylinder multi-field	1	D1	31.54
cylinder scalar	1	D1	31.83
cylinder multi-field	6	C64-C64-D256-D256	24.64
cylinder scalar	6	C64-C64-D256-D256	45.42
cylinder multi-field	6	D64	43.97
cylinder scalar	6	D64	53.59
cylinder multi-field	6	D1	167.03
cylinder scalar	6	D1	176.27

TABLE 1: Comparison of model error using scalar and multivariate prediction. Multivariate input data leads to fewer errors, especially for larger models and far-ahead prediction, since they have more capacity to leverage the additional information.

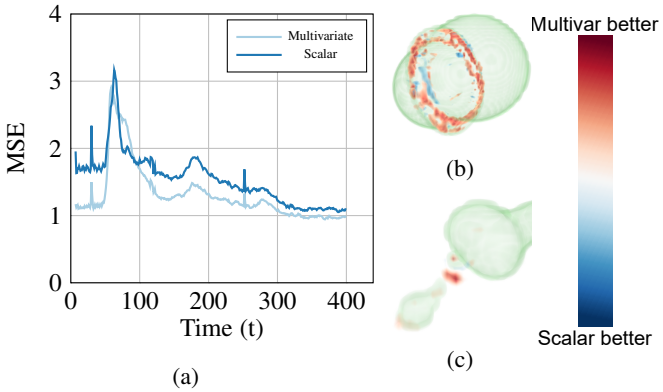


Fig. 18: Comparison of scalar and multivariate prediction on "droplet". **a:** Temporal misprediction views: scalar prediction is less accurate overall, but detects similar temporal events. **b, c:** Spatial difference between models' error regions, with **red** showing where the scalar model had larger errors, and **blue** – the opposite. The multivariate model is able to better predict the disk expansion (**b**) and small droplet separation (**c**) events.

We performed a set of experiments on one of the turbulent CFD ensemble members. The model was supplied with velocity vectors, velocity magnitude and pressure fields, while predicting the velocity magnitude. We used three models of different capacity ("C64-C64-D256-D256", "D64-D32" and "D1"), two prediction delays (one and six frames) and compared to predictions using only the scalar velocity magnitude field as input (Table 1). As we can see, providing additional fields to the model results in lower overall prediction error. The differences are more significant when we increase the delay: predicting further into the future is easier with the additional information. We also notice that larger models benefit more from the additional fields, since they can represent a more complex relationship between the input fields.

To investigate the differences in prediction error regions between the multifield and scalar models we performed an additional experiment on the droplet dataset. We compared a multifield model that received volume-of-fluid, pressure and velocity vector fields, to a scalar model that received only the volume-of-fluid values. The multifield model achieved an overall MSE of 0.14 and the scalar model an MSE of 0.23 (values are relatively low due to large amounts of empty space in the dataset). Thus, similarly to the previous experiments, providing more data allowed for an overall more accurate prediction. In Fig. 18a we present the temporal misprediction of both models, where we see that both highlight the same events, with the only significant difference

occurring around the initial droplet collision (frames 50-100). The multivariate model has lower errors overall, though the scalar model "recovers" slightly faster due to focusing on the motion of fluid and being unaffected by changes within other fields. In Fig. 18b we show the difference between the models' absolute errors at frame 62 around the initial collision. We see that the scalar model produces larger errors around the expanding disk between two droplets, while the multivariate model is able to use the high pressure values that accompany this expansion to distinguish and more accurately predict this behavior. Then, we inspected the data for other regions of significant deviation and found several spatially-smaller regions near ligament separations, where the multivariate model produces a more accurate prediction (Fig. 18c).

Overall, we found that although multifield and scalar models are generally similar, providing and withholding fields from the model can in some cases affect the results qualitatively. Both approaches can be used depending on the application scenario, while combining them can act as an additional analysis tool.

8.7 Performance

We provide performance measurements of our implementation in Table 2. All tests have been performed on a machine with an Intel Xeon E5-2630 v4 CPU and an Nvidia Tesla V100 GPU. The dataset size is provided in the format Time $\times$ Width $\times$ Height $\times$ Depth, and p_u refers to the undersampling probability (Sec. 3.3). We also provide the number of training epochs that passed before the convergence condition was triggered (Sec. 7). We used separate configurations for 2D and 3D datasets, with stronger undersampling and a smaller patch size for the 3D datasets, which otherwise generate superfluous amounts of data for our relatively simple models. Overall we can see that prediction often takes a considerable amount of the execution time. This is due to the fact that we cannot undersample the prediction data and need to obtain a prediction for each data voxel. The prediction patches do not fit into RAM and have to be extracted out-of-core, triggering expensive reads from the hard drive. For datasets with large amount of empty space (e.g. droplet) we can reduce the amount of processing (see Sec. 7), while others don't allow this optimization.

For our parameter selection study (Sec. 8.5) on the "vortex street" dataset we performed 80 runs, spending a total of 64 hours to train the models and 108 minutes to perform the predictions. For the "hotroom" dataset (500x181x91x181 in size) we used stronger undersampling ($p_u = 10^{-4}$) to perform 64 runs with a total of 54 hours to train and 56 hours to predict. Although the time required to sample the parameter space is significant, the task is highly parallelizable and has linear speedup, since each experiment can be performed on a separate machine.

Dataset	Dataset size	Patch size	Train set	p_u	Model*	Train (min.)	Predict (min.)	Epoch
droplet	400x256x256x256	$5x5^3$	712.1K	0.01	A	35.0	171.5	76
droplet	400x256x256x256	$5x5^3$	712.1K	0.01	B	79.6	170.7	173
droplet	400x256x256x256	$5x5^3$	714.0K	0.01	C	135.3	177.6	315
spheres	200x128x128x128	$5x5^3$	185.2K	0.01	A	7.3	16.6	60
spheres	200x128x128x128	$5x5^3$	185.2K	0.01	B	56.7	16.3	500
spheres	200x128x128x128	$5x5^3$	185.4K	0.01	C	16.5	16.3	146
bottle	465x450x215x1	$15x5^2$	3.4M	0.1	A	87.9	8.8	71
bottle	465x450x215x1	$15x5^2$	3.4M	0.1	B	234.2	8.6	200
bottle	465x450x215x1	$15x5^2$	3.4M	0.1	C	26.2	10.3	23
vortex st.	800x101x301x1	$15x5^2$	1.8M	0.1	A	115.9	5.8	173
vortex st.	800x101x301x1	$15x5^2$	1.8M	0.1	B	157.3	5.8	250
vortex st.	800x101x301x1	$15x5^2$	1.8M	0.1	C	31.4	5.6	52

TABLE 2: Performance of our local prediction implementation with different datasets and models. We undersample the training data with probability p_u and cache prediction results for empty space. Large and dense 3D volumes require the most computation. * *Model A*: C64-C64-D256-D256, *Model B*: D256, *Model C*: D1.

9 Discussion

Local prediction. Our choice of making the prediction based on local information yields several advantages, including training performance, translation-invariance and model simplicity (Sec. 3.3). It also introduces an additional parameter – the patch size, which can have an impact on prediction performance both in terms of execution time and accuracy (Sec. 8.4.2). Although locality is often a reasonable assumption for scientific data, relationships extending beyond the patch size cannot be captured by our models. When such effects need to be accounted for, the models could be extended to incorporate global context, e.g., by using downsampled data.

Performance. A limiting factor of our approach is performance. The performance costs come from training multiple neural networks on large data, as well as evaluating them to obtain predictions (Table 2). Although smaller patch sizes combined with undersampling are often sufficient to produce accurate results, analyzing even moderately-sized datasets may take hours and days. Fortunately, the approach has large parallelization potential. Both neural network training and prediction can be distributed in terms of data across multiple machines, and parameter studies are even easier to parallelize. Pretrained and reused models can be investigated for applications within the same domain or simulation ensembles (cf. Sec. 8.3). Finally, a lot of effort is currently put into development of specialized ML hardware architectures, which may provide further acceleration benefits in the future.

Temporal interpolation Our approach operates by detecting regions that are hard to predict based on past events. We have shown that by varying the prediction delay we can vary the difficulty of prediction, increasing it for datasets with fine temporal resolution (Sec. 8.4.1). However, this also implies that in the opposite case of very coarse temporal resolution the prediction task can become too hard, causing the models to produce high errors everywhere, making it impossible to distinguish behavior of different complexity. An interesting extension to address this case could be to simplify the prediction problem by considering not only the past, but the future too, essentially performing interpolation, which would have lower resolution requirements than extrapolation.

Sequence modeling. For simplicity and generality we used relatively basic neural network models (Sec. 3.2, Sec. 7). However, building upon our findings with simpler models, more complex and specialized architectures could be used to improve the results. For example, recurrent neural networks are being successfully used to model temporal data and might provide a more meaningful prediction for spatiotemporal volumes as well. Moreover, they can be applied to temporal sequences of variable length, which can produce additional information about the local temporal complexity via monitoring the length of accurately predicted sequences. Similar information could also be obtained with current models by performing prediction on the results of past predictions and observing how fast the error accumulates.

Learning physics. An interesting question that comes up in the context of applying neural networks to scientific data is whether the network learns the underlying physical laws. Conceptually, when doing prediction on simulation data one would expect the future values to be completely predictable, given that all relevant fields

are available. After all, the values are computed by a numerical solver using deterministic principles. Furthermore, the exact update rules used during integration are often rather simple functions of local information (possibly even linear, e.g. for diffusion). Thus, our models should perform almost-perfect prediction, resulting in no significant errors and detecting no irregular regions.

However, as demonstrated by the cross-prediction performance on ensemble data (Sec. 8.3), learning to accurately predict performance of one simulation run might still produce significant errors on a different run, although the respective simulation code has not changed. It appears that rather than learning the underlying physics, the model efficiently represents, stores and interpolates the observed behavior. But this presents an advantage for visualization: by capturing different local scenarios the model allows us to distinguish different processes in the data, helping us to analyze the produced results, rather than the underlying physical principles.

Nevertheless, to validate the conceptual possibility of a "perfect" prediction for simulation data, we performed an experiment on one of the turbulent ensemble members (used in Sec. 8.6), but with all the solver timesteps being exported, increasing the number of timesteps from 54 to 32000. Using the linear "D1" model we achieved an MSE of 0.01, i.e. a practically perfect prediction (cf. Table 1). This shows that given exhaustive information the underlying simulation rules can be learned for a given dataset.

In practice however, the simulation codes rarely export all the values computed at runtime (cf. in situ visualization), with a single data timestep often corresponding to hundreds of solver steps. We can also control this using the prediction delay parameter (Sec. 3.3). Multiple simple updates to interacting variables can compound to complex functions, making the prediction significantly harder in some regions and allowing us to distinguish different behavior in visualization.

Physical constraints. While we have used models with no physical or domain-specific assumptions, there is a number of generic physical properties that could be explicitly modeled: energy and mass conservation, flow incompressibility, etc. These could be introduced to the model's loss function to aid in doing physically-accurate predictions, or to find spatiotemporal regions where the model violates these conditions.

Enhancing user control. Currently the user can influence the results by defining models and some parameters before running our method, and explore the results interactively afterwards (Sec. 8.1). However, our approach could be extended to provide further control to the analyst. On the one hand, ML-based approaches like one-shot learning could be explored, reusing the feature space learned by the models to detect more specific behavior based on the analyst's feedback. On the other hand, more traditional visualization techniques like clustering can be applied to discover further instances of similar behavior or filter detected irregularities.

10 Conclusion

In this paper we presented our approach to detecting regions of irregular behavior in spatiotemporal data. We used an ensemble of machine learning models to perform predictions of local behavior, quantifying the resulting error and using it as an indicator of complexity. The resulting misprediction volumes were then used to construct aggregated spatial and temporal views of high-error regions, effectively focusing on unpredictable events and filtering out trivial behavior. We demonstrated that models of varying capacity have different "sensitivity" to irregular behavior, highlighting areas with different degrees of irregularities. By using generic neural network architectures and no manually-defined features, we were able to detect and visualize meaningful spatiotemporal events in the data without making any dataset- or domain-specific assumptions.

We also introduced two additional applications of local prediction models. Based on the analysis of temporal irregularity detected by several models, we performed adaptive timestep selection, picking timesteps in which significant events and transitions occurred, while avoiding repetitive periodic behavior. Furthermore, we proposed a cross-prediction-based dissimilarity measure for the analysis of ensemble data, showing that it captures expressive qualitative differences in flow simulation behavior.

Finally, after reviewing the current limitations, we discussed future extensions, including parallelization, modeling of physical assumptions and improvements in user interaction.

Acknowledgments

This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2075 (SimTech) – 390740016.

References

[1] A. Joshi and P. Rheingans, "Illustration-inspired techniques for visualizing time-varying data," in *Visualization, 2005. VIS 05. IEEE*, 2005, pp. 679–686.

[2] A. Lu and H.-W. Shen, "Interactive storyboard for overall time-varying data visualization," in *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific*, 2008, pp. 143–150.

[3] X. Tong, T.-Y. Lee, and H.-W. Shen, "Salient time steps selection from large scale time-varying data sets with dynamic time warping," in *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on*, 2012, pp. 49–56.

[4] S. Frey and T. Ertl, "Flow-based temporal selection for interactive volume visualization," *Computer Graphics Forum*, 2016.

[5] J.-P. Balabanian, I. Viola, T. Möller, and E. Gröller, "Temporal styles for time-varying volume data," in *Proceedings of 3DPVT'08 - the Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, S. Gumhold, J. Kosecka, and O. Staadt, Eds., June 2008, pp. 81–89.

[6] L. Liu, D. Silver, K. Bemis, D. Kang, and E. Curchitser, "Illustrative visualization of mesoscale ocean eddies," *Computer Graphics Forum*, vol. 36, no. 3, pp. 447–458, 2017.

[7] Z. Fang, T. Möller, G. Hamarneh, and A. Celler, "Visualization and exploration of time-varying medical image data sets," in *Proceedings of Graphics Interface 2007*, ser. GI '07. New York, NY, USA: ACM, 2007, pp. 281–288.

[8] T.-Y. Lee and H.-W. Shen, "Visualizing time-varying features with tac-based distance fields," in *Visualization Symposium, 2009. PacificVis '09. IEEE Pacific*, 2009, pp. 1–8.

[9] ——, "Visualization and exploration of temporal trend relationships in multivariate time-varying data," *IEEE Vis. Comput. Gr.*, vol. 15, no. 6, pp. 1359–1366, 2009.

[10] C. Wang, H. Yu, and K.-L. Ma, "Importance-driven time-varying data visualization," *IEEE Vis. Comput. Gr.*, vol. 14, no. 6, pp. 1547–1554, 2008.

[11] S. Frey, F. Sadlo, and T. Ertl, "Visualization of temporal similarity in field data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, pp. 2023–2032, 2012.

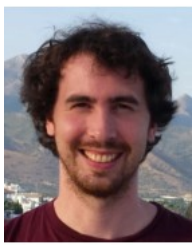
[12] S. Dutta and H. W. Shen, "Distribution Driven Extraction and Tracking of Features for Time-varying Data Analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 22, no. 1, pp. 837–846, Jan. 2016.

[13] F.-Y. Tzeng and K.-L. Ma, "Intelligent Feature Extraction and Tracking for Visualizing Large-Scale 4D Flow Simulations," in *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference*, Nov. 2005, pp. 6–6.

[14] C. Muelder and K. L. Ma, "Interactive feature extraction and tracking by utilizing region coherency," in *2009 IEEE Pacific Visualization Symposium*, Apr. 2009, pp. 17–24.

[15] J. Woodring, C. Wang, and H.-W. Shen, "High dimensional direct rendering of time-varying volumetric data," in *Visualization, 2003. VIS 2003. IEEE*, 2003, pp. 417–424.
[16] B. Bach, P. Dragicevic, D. Archambault, C. Hurter, and S. Carpendale, "A descriptive framework for temporal data visualizations based on generalized space-time cubes," *Comput. Graph. Forum*, 2016.
[17] S. Frey and T. Ertl, "Progressive direct volume-to-volume transformation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 921–930, Jan 2017.
[18] S. Frey, "Sampling and estimation of pairwise similarity in spatio-temporal data based on neural networks," *Informatics*, vol. 4, no. 3, 2017.
[19] U. Bordoloi and H.-W. Shen, "View Selection for Volume Rendering," in *Proceedings of the IEEE Visualization Conference*, vol. 62, Nov. 2005, pp. 487–494.
[20] I. Viola, M. Feixas, M. Sbert, and M. E. Groller, "Importance-Driven Focus of Attention," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 933–940, Sep. 2006.
[21] M. Chen and H. Jaenicke, "An Information-theoretic Framework for Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1206–1215, Nov. 2010.
[22] Y. Wang, W. Chen, J. Zhang, T. Dong, G. Shan, and X. Chi, "Efficient Volume Exploration Using the Gaussian Mixture Model," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 11, pp. 1560–1573, Nov. 2011.
[23] H. Jänicke, A. Wiebel, G. Scheuermann, and W. Kollmann, "Multifield visualization using local statistical complexity," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1384–1391, Nov. 2007.
[24] K. L. Ma, "Machine learning to boost the next generation of visualization technology," *IEEE Computer Graphics and Applications*, vol. 27, no. 5, pp. 6–9, Sept 2007.
[25] A. Endert, W. Ribarsky, C. Turkay, B. Wong, I. Nabney, I. Díaz Blanco, and F. Rossi, "The state of the art in integrating machine learning into visual analytics," 3 2017.
[26] R. Fuchs, J. Waser, and M. E. Groller, "Visual Human+Machine Learning," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, pp. 1327–1334, Nov. 2009.
[27] P. Klemm, S. Saalfeld, K. Lawonn, M. Rak, H. Völzke, K. Hegenscheid, and B. Preim, "Interactive Visual Analysis of Lumbar Back Pain What the Lumbar Spine Tells About Your Life," in *IVAPP 2015 - 6th International Conference on Information Visualization Theory and Applications; VISIGRAPP, Proceedings*, Mar. 2015.
[28] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.
[29] G. Andrienko, N. Andrienko, S. Bremm, T. Schreck, T. von Landesberger, P. Bak, and D. Keim, "Space-in-time and time-in-space self-organizing maps for exploring spatiotemporal patterns," in *Proceedings of the 12th Eurographics / IEEE - VGTC Conference on Visualization*, ser. EuroVis'10. Chichester, UK: The Eurographs Association; John Wiley; Sons, Ltd., 2010, pp. 913–922. [Online]. Available: http://dx.doi.org/10.1111/j.1467-8659.2009.01664.x
[30] D. Sacha, M. Kraus, J. Bernard, M. Behrisch, T. Schreck, Y. Asano, and D. A. Keim, "Somflow: Guided exploratory cluster analysis with self-organizing maps and analytic provenance," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 120–130, Jan 2018.
[31] M. Berger, J. Li, and J. A. Levine, "A Generative Model for Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 4, pp. 1636–1650, Apr. 2019.
[32] J. Han, J. Tao, and C. Wang, "FlowNet: A Deep Learning Framework for Clustering and Selection of Streamlines and Stream Surfaces," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2018.
[33] A. Hassanien, M. Elgharib, A. Selim, S.-H. Bae, M. Hefeeda, and W. Matusik, "Large-scale, Fast and Accurate Shot Boundary Detection through Spatio-temporal Convolutional Neural Networks," *arXiv:1705.03281 [cs]*, May 2017.
[34] M. Gygli, "Ridiculously Fast Shot Boundary Detection with Fully Convolutional Neural Networks," *arXiv:1705.08214 [cs]*, May 2017.
[35] S. R. Kulkarni and G. Harman, "Statistical learning theory: A tutorial," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 3, no. 6, pp. 543–556, Nov. 2011.
[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[37] V. N. Vapnik and Chervonenkis, A. Ya., "On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities," *Theory of Probability & Its Applications*, vol. 16, no. 2, May 1969.
[38] H. Obermaier and K. I. Joy, "Future challenges for ensemble visualization," *IEEE Computer Graphics and Applications*, vol. 34, no. 3, pp. 8–11, May 2014.
[39] K. Potter, A. Wilson, P.-T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johnson, "Ensemble-Vis: A Framework for the Statistical Visualization of Ensemble Data," in *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*, ser. ICDMW '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 233–240.
[40] J. Waser, R. Fuchs, H. Ribicic, B. Schindler, G. Bloschl, and E. Groller, "World Lines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1458–1467, Nov. 2010.
[41] S. Bruckner and T. Moller, "Result-Driven Exploration of Simulation Parameter Spaces for Visual Effects Design," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 1468–1476, Nov. 2010.
[42] M. Hummel, H. Obermaier, C. Garth, and K. I. Joy, "Comparative Visual Analysis of Lagrangian Transport in CFD Ensembles," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2743–2752, Dec. 2013.
[43] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Möller, "Visual Parameter Space Analysis: A Conceptual Framework," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 2161–2170, Dec. 2014.
[44] Z. Bar-Joseph, D. K. Gifford, and T. S. Jaakkola, "Fast optimal leaf ordering for hierarchical clustering," in *Intelligent Systems in Molecular Biology*, vol. 17, Jun. 2001, pp. 22–29.
[45] A. Velten, D. Wu, A. Jarabo, B. Masia, C. Barsi, C. Joshi, E. Lawson, M. Bawendi, D. Gutierrez, and R. Raskar, "Femto-photography: Capturing and visualizing the propagation of light," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 44:1–44:8, 2013.
[46] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Dec. 2014.

Gleb Tkachev received his Masters degree in computer science from the University of Stuttgart, Germany. He is a PhD student at the University of Stuttgart Visualization Research Center (VISUS). His current research interests are focused on combining visual computing and machine learning methods for the analysis of scientific data.



Steffen Frey received his PhD degree in computer science from the University of Stuttgart, Germany. Currently, he is a PostDoc at the University of Stuttgart Visualization Research Center (VISUS). His research interests are in visual analysis techniques for large and complex data in scientific visualization, with a particular focus on performance-related aspects and expressive visual representations of dynamic processes.

Thomas Ertl received the MS degree in computer science from the University of Colorado at Boulder and the PhD degree in theoretical astrophysics from the University of Tübingen. He is a full professor of computer science with the University of Stuttgart, Germany in the Visualization and Interactive Systems Institute (VIS) and the director of the Visualization Research Center (VISUS). His research interests include visualization, computer graphics, and human computer interaction.