TSR-TVD: Temporal Super-Resolution for Time-Varying Data Analysis and Visualization

Jun Han and Chaoli Wang, *Senior Member, IEEE*

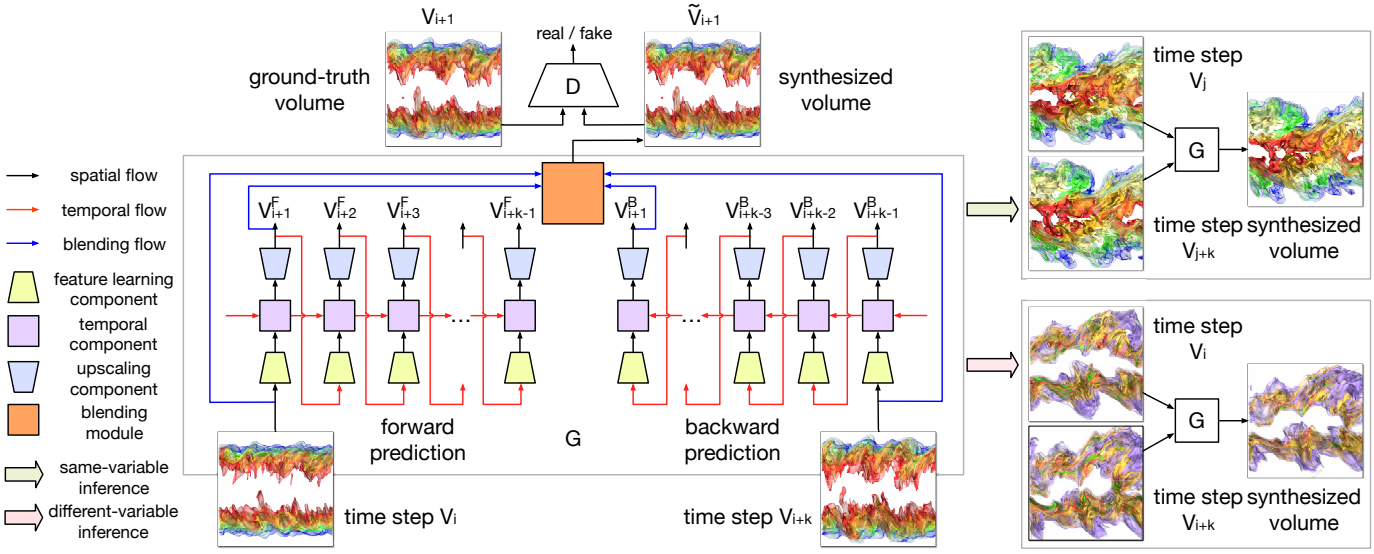


Fig. 1. TSR-TVD learns the generation of intermediate volumes from a given pair of volumes. The network includes a generator G which consists of the predicting and blending modules and a discriminator D which distinguishes the synthesized volumes from the ground truth (GT) volumes. Once learned, the network can perform both same-variable and different-variable inferences using G.

Abstract— We present TSR-TVD, a novel deep learning framework that generates temporal super-resolution (TSR) of time-varying data (TVD) using adversarial learning. TSR-TVD is the first work that applies the recurrent generative network (RGN), a combination of the recurrent neural network (RNN) and generative adversarial network (GAN), to generate temporal high-resolution volume sequences from low-resolution ones. The design of TSR-TVD includes a generator and a discriminator. The generator takes a pair of volumes as input and outputs the synthesized intermediate volume sequence through forward and backward predictions. The discriminator takes the synthesized intermediate volumes as input and produces a score indicating the realness of the volumes. Our method handles multivariate data as well where the trained network from one variable is applied to generate TSR for another variable. To demonstrate the effectiveness of TSR-TVD, we show quantitative and qualitative results with several time-varying multivariate data sets and compare our method against standard linear interpolation and solutions solely based on RNN or CNN.

Index Terms—Time-varying data visualization, super-resolution, deep learning, recurrent generative network

1 Introduction

In many fields of science, scientists run large-scale scientific simulations and produce time-varying multivariate data on a daily basis. In this paper, we focus on augmenting the temporal resolution of these data as scientists often simulate a long temporal sequence with many variables involved but could only afford to store a very limited number of time steps (e.g., every hundredth time step) for post-hoc analysis. Our goal is to augment these reduced simulation data during post-processing by generating the refined temporal resolution to enable more accurate investigation of dynamic spatiotemporal features of the underlying data. That is, given a low-resolution volume sequence of, for example, 100 time steps, we aim to generate the high-resolution volume sequence of, for example, 500 time steps, while keeping the spatial resolution intact.

Producing temporally refined volume sequences for time-varying data poses two main challenges. The first challenge is that the dynamic change of volume data over time is typically non-linear. Conventional approaches employ standard linear interpolation (LERP) to generate intermediate volumes. These interpolations are only based on local information around the interpolated position, and therefore may not capture the complex evolution and non-linear changes of volumes. The second challenge is how to take visual quality into consideration. Applying a typical recurrence-based neural network design to our problem would only measure the voxelwise distance, which may not lead to high-quality rendering results even though the PSNR is high. For example, Zhang et al. [61] pointed out that only using a pixelwise loss function to train a neural network could generate images with noise and artifacts.

To address these challenges, we present TSR-TVD, a novel deep learning framework for producing temporal super-resolution (TSR)

• *J. Han and C. Wang are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556. E-mail: {jhan5, chaoli.wang}@nd.edu.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

from time-varying data (TVD). We leverage the *recurrent generative network* (RGN), a combination of the *recurrent neural network* (RNN) and *generative adversarial network* (GAN), to achieve TSR. This is because RNN and GAN can learn the temporal and spatial relationship among different volumes non-uniformly and non-locally, thus interpolating the intermediate volumes with high quality. Inspired by the sequence learning (e.g., weather forecasting and machine translation) and image generation (e.g., frame interpolation and video prediction) techniques, our solution consists of a generator and a discriminator for producing temporally coherent TSR of a sequence of volumes using adversarial learning. TSR-TVD takes the sampled volumes as input and generates the intermediate volumes. We train TSR-TVD by optimizing the loss function that includes adversarial loss, volumetric loss, and feature loss. Our TSR-TVD handles multivariate data as well where the trained network from one variable is applied to generate TSR for another variable. To demonstrate the effectiveness of TSR-TVD, we show quantitative and qualitative results with several time-varying multivariate data sets of different characteristics. We compare TSR-TVD against the widely-used LERP and solutions based on RNN or CNN only. We show that our method achieves a better quality than LERP in terms of PSNR at the data level and the best quality in terms of SSIM at the image level.

The contributions of this paper are as follows. First, our work is the first that applies RGN (a combination of RNN and GAN) for generating TSR of a volume sequence. Second, we apply ConvLSTM layers for capturing spatiotemporal relationships and propose a new voxel shuffle layer for accelerating the training process. Third, we design a new architecture for the TSR task, which supports not only same-variable inference but also different-variable inference. Fourth, we investigate several hyperparameter settings and analyze how they impact the performance of TSR-TVD.

2 Related Work

Time-varying multivariate data analysis and visualization is a key topic in scientific visualization. We refer interested readers to the survey paper [24] for an overview. Existing works on time-varying data analysis and visualization focus on efficient organization and rendering [44, 52, 59], transfer function specification [20], illustration-inspired [23] and importance-driven [54] techniques. For time-varying multivariate data, researchers have investigated query-driven visualization [47, 9], exploration of variable correlation [43], variable grouping [4], and information flow between variables [53, 29], as well as navigation interfaces [2, 48]. In the following, we restrict our attention to related work on temporal super-resolution for video, relevant GAN and RNN techniques, and deep learning for scientific visualization.

TSR for video. Deep learning has achieved great success in generating TSR for video. For instance, Niklaus et al. [35] introduced a *convolutional neural network* (CNN) for frame interpolation where the network learns a kernel through the input frames and then applies the learned kernel to generate the missing frames. Nguyen et al. [34] proposed a deep linear embedding model to interpolate the intermediate frames. They transformed each frame into a feature space, then linearly interpolated the intermediate frames in the feature space, and finally recovered the interpolated features to the corresponding frames. Jiang et al. [21] established a CNN to estimate the forward and backward optical flows via two given frames. They then wrapped these optical flows into the frames to generate arbitrary in-between frames.

Our work differs from the above works. First, we leverage RNN to capture the temporal relationship rather than using CNN, since RNN can learn the potential pattern across time. Second, unlike image generation tasks where researchers can use a pre-trained image classification model to improve the visual quality of the synthesized results, there is no such model for volumes. Third, many deep learning models compute the optical flow to generate intermediate frames but these models need either a lot of labeled optical flow data for training or an additional subnetwork to calculate the optical flow in an unsupervised way. Therefore, We apply GAN to ensure the high quality of the synthesized volumes. There are works that use RNN or GAN for similar purposes [18, 30, 6], but not at the same time as our work does. Our approach does not require a lot of labeled data or additional modules, which reduces its complexity and training time. We incorporate ConvLSTM to achieve temporal coherence. Based on this mechanism, TSR-TVD uses all information from time steps $i, i+1, \cdots, i+j-1$ to interpolate time step $i+j$.

GAN and RNN techniques. Introduced by Goodfellow et al. [10], a GAN includes two networks: a generator G and a discriminator D. G tries to synthesize data samples from noise or observation to fool D, while D aims to distinguish the data generated by G from the real samples. GAN has been applied to image translation [19, 55], style transfer [22], image inpainting [37], and image super-resolution [28] tasks. Due to the problem of unstable training, different techniques have been considered during training, such as L_2 loss [37], gradient regularization [32], and separate learning rates for G and D [42]. An RNN accepts a sequence as input and builds the temporal relationships through the internal states. One of the most frequently used RNN architectures is *long short-term memory* (LSTM) [15], which has been applied to video colorization [27] and video caption [51]. Due to the problem of gradient vanishing and explosion, different techniques have been applied in RNN training, such as stacked LSTMs [51] and gradient clipping [36].

Deep learning for scientific visualization. There is a growing body of works that apply deep learning to solve scientific visualization problems. Tzeng et al. [49, 50] pioneered the use of artificial neural networks for classifying 3D volumetric data sets. Ma [31] pointed out the use of neural networks as a promising direction for visualization research. With the explosive growth of modern deep learning techniques, researchers have recently started to explore the capabilities of *deep neural network* (DNN) to address various problems.

For volume visualization, Zhou et al. [62] presented a CNN-based solution for volume upscaling which better preserves structural details and volume quality than linear upscaling. Raji et al. [39] leveraged CNNs to iteratively refine a transfer function, aiming to match the visual features in the rendered image of a similar volume data set with the one in the target image. Cheng et al. [7] presented a deep-learning-assisted solution which depicts and explores complex structures that are difficult to capture using conventional approaches. Berger et al. [3] designed a GAN to compute a model from a large collection of volume-rendering images conditioned on viewpoints and transfer functions. Shi and Tao [45] proposed a CNN-based viewpoint estimation method that achieves good performance on images rendered with different transfer functions and rendering parameters. Xie et al. [60] designed a temporally coherent approach to generate spatial super-resolution volumes where temporal coherence is guaranteed through a temporal discriminator. Weiss et al. [57] presented an image-space solution that learns to upscale a sampled representation of geometric properties of an isosurface at low resolution to a higher resolution. They considered temporal variations by adding a frame-to-frame motion loss to achieve improved temporal coherence.

For flow visualization, Hong et al. [16] used LSTM to learn and predict data access patterns in particle tracing in order to hide the I/O latency in distributed and parallel flow visualization. Wiewel et al. [58] took a LSTM-based approach to predict dense 3D+time functions of physics system using an autoencoder framework. Kim and Günther [25] combined filter and feature extraction in an end-to-end manner using CNN for robust reference frame extraction from unsteady 2D vector fields. Han et al. [12] proposed a deep learning method for vector field reconstruction that takes the streamlines traced from the original vector fields as input and applies a two-stage process to reconstruct high-quality vector fields. Han et al. [11] designed an autoencoder to learn the features of flow lines or surfaces in the latent space and performed dimensionality reduction and interactive clustering for representative selection.

Although generating a spatial high-resolution volume from a low-resolution one has been studied [62, 60], to our best knowledge, in the context of time-varying data analysis and visualization, no work has been done that generates a temporal high-resolution volume sequence from a low-resolution one, which is the focus of this work.

3 Our Recurrent Generative Approach

We define the TSR problem as follows. Given a pair of volumes (V_i, V_{i+k}) from time steps i and $i+k$ (where $k>1$), we seek a function ϕ that satisfies $\phi(V_i, V_{i+k}) \approx \mathbf{V}$, where $\mathbf{V} = \{V_{i+1}, V_{i+2}, \cdots, V_{i+k-1}\}$ are the intermediate volumes between V_i and V_{i+k}.

We propose a novel *recurrent generative network* (RGN) which is a combination of RNN and GAN. The RGN includes a generator G and a discriminator D. G uses two modules to estimate the function ϕ, as sketched in Figure 1. The first module, the *predicting module* (ϕ_{PREDICT}), is a volume prediction network that produces a forward prediction $\mathbf{V}^{\text{F}}$ through V_i and a backward prediction $\mathbf{V}^{\text{B}}$ through V_{i+k}, respectively. Namely,

$$\mathbf{V}^{\text{F}} = \phi^{\text{F}}_{\text{PREDICT}}(V_i), \tag{1}$$

$$\mathbf{V}^{\text{B}} = \phi^{\text{B}}_{\text{PREDICT}}(V_{i+k}), \tag{2}$$

$$\phi_{\text{PREDICT}} = \{\phi^{\text{F}}_{\text{PREDICT}}, \phi^{\text{B}}_{\text{PREDICT}}\}. \tag{3}$$

The second module, the *blending module* (ϕ_{BLEND}), takes V_i, V_{i+k}, and the corresponding pair of volumes from $\mathbf{V}^{\text{F}}$ and $\mathbf{V}^{\text{B}}$ that share the same time step (for clarity, Figure 1 only illustrates the blending flow connections with time step $i+1$) as input, and blends them into a volume that represents the final prediction $\tilde{\mathbf{V}}$, i.e.,

$$\tilde{\mathbf{V}} = \phi_{\text{BLEND}}(V_i, V_{i+k}, \mathbf{V}^{\text{F}}, \mathbf{V}^{\text{B}}). \tag{4}$$

As we use the down-sampling and up-sampling framework to reconstruct intermediate volumes, detailed features may be lost during down-sampling which cannot be recovered perfectly during up-sampling. Adding V_i and V_{i+k} into the blending will help to eliminate noise generated from the predictions. The discriminator D distinguishes $\tilde{\mathbf{V}}$ from $\mathbf{V}$. That is, given different inputs to D, D outputs a score to indicate the realness of the input. Ideally, $D(\mathbf{V}) \approx 1$ and $D(\tilde{\mathbf{V}}) \approx 0$. In this regard, D can be treated as a binary classifier. The score from D can guide G in synthesizing high-quality volumes since the goal of G is to fool D so that D cannot distinguish $\tilde{\mathbf{V}}$ as fake volumes.

In the following, we describe the details of TSR-TVD, including the definition of the loss function and the architectures of G and D. The training algorithm and optimization details can be found in Section 1 in the Appendix.

3.1 Loss Function

Notations. Let us denote $\mathbf{V}^T = \{(V_1, V_{k_1}), (V_{k_1}, V_{k_2}), \cdots, (V_{k_{n-1}}, V_{k_n})\}$ as a set of input volume pairs for our TSR-TVD framework, and $\mathbf{V}^I = \{\{V_2, \cdots, V_{k_1-1}\}, \{V_{k_1+1}, \cdots, V_{k_2-1}\}, \cdots, \{V_{k_{n-1}+1}, \cdots, V_{k_n-1}\}\}$ as the ground truth (GT) intermediate volumes that we aim to interpolate. Let θ_G and θ_D be the learnable parameters in G and D, respectively, and K be the maximal interpolation step (i.e., $k_{i+1} - k_i \leqslant K+1, i \in [0, n-1]$).

Adversarial loss. Following the definition in GAN [10], we define the adversarial loss for G and D as follows

$$\min_{\theta_G} \mathscr{L}_G = \mathbb{E}_{V \in \mathbf{V}^T}\big[\log D(G(V))\big], \tag{5}$$

$$\min_{\theta_D} \mathscr{L}_D = \frac{1}{2}\mathbb{E}_{V \in \mathbf{V}^I}\big[\log D(V)\big] + \frac{1}{2}\mathbb{E}_{V \in \mathbf{V}^T}\big[\log\big(1 - D(G(V))\big)\big], \tag{6}$$

where $\mathbb{E}[\cdot]$ denotes the expectation operation.

The idea behind this formulation is that we let G and D compete with each other so that G can produce synthesized volumes which fully fool D as the training goes, while the goal of D is to distinguish the synthesized volumes as fake and return this information to G to force G to be more powerful until D and G achieve a balance. Through this process, G can learn how to synthesize high-quality volumes that are highly close to the GT. This competition idea behind adversarial loss encourages a perceptual solution rather than a PSNR or SSIM orientated solution by minimizing voxelwise loss, such as L_1 loss. Note that using the adversarial loss alone would lead to an unstable training for G and D. Therefore, we further consider volumetric loss and feature loss, which are described next.

Volumetric loss. Since the adversarial loss only considers perceptual results, we add a volumetric loss in $\mathscr{L}_G$, which means that the task of G is to not only fool D but also get close to the GT output in the L_2 sense, while the job of D remains unchanged. In addition, the training stability can also be improved through adding the volumetric loss [37, 19]. Therefore, we utilize the L_2 distance as a part of our loss function for G

$$\mathscr{L}_V = \mathbb{E}_{V' \in \mathbf{V}^T, V \in \mathbf{V}^I}\big[||G(V') - V||_2\big], \tag{7}$$

where $||\cdot||_2$ denotes the L_2 norm.

Feature loss. Zhang et al. [61] suggested that features extracted from DNNs can be used as a metric to evaluate the output of generative models. Following this guideline, we also compute the feature difference between $\mathbf{V}^I$ and $G(\mathbf{V}^T)$ based on D. This feature loss enforces G to produce similar features between $G(\mathbf{V}^T)$ and $\mathbf{V}^I$ at different scales. Specifically, the features are extracted from every convolutional (Conv) layers of D except the final Conv layer and G can try to minimize these intermediate representations between the real and synthesized volumes. We denote the feature representation extracted from the kth Conv layer as F^k. Then the feature loss is defined as

$$\mathscr{L}_F = \mathbb{E}_{V' \in \mathbf{V}^T, V \in \mathbf{V}^I} \sum_{k=1}^{N-1} \frac{1}{N_k}\big[||F^k(G(V')) - F^k(V)||_2\big], \tag{8}$$

where N is the total number of Conv layers in D and N_k denotes the number of elements in the kth Conv layer. This feature loss is related to the VGG loss [22, 55] which has achieved impressive results in image super-resolution and style transfer tasks.

Taking all three losses into consideration, we define the final loss function for G as

$$\min_{\theta_G} \mathscr{L}_G = \lambda_1\big(\mathbb{E}_{V \in \mathbf{V}^T}\big[\big(D(G(V)) - 1\big)^2\big]\big) + \lambda_2 \mathscr{L}_V + \lambda_3 \mathscr{L}_F, \tag{9}$$

where λ_1, λ_2, and λ_3 are hyperparameters, which control the relative importance of these three losses.

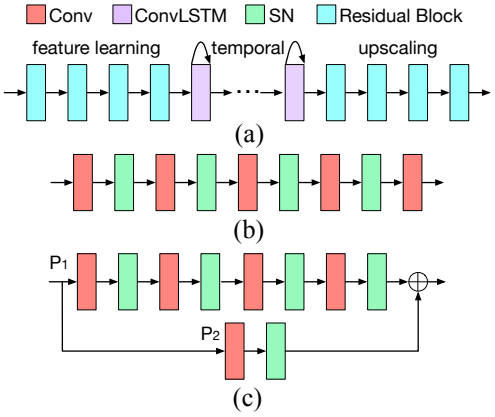


Fig. 2. Network architecture of (a) the predicting module in generator G and (b) discriminator D. (c) An example of the residual block.

3.2 Network Architecture

Generator. As sketched in Figure 1, the generator G contains two modules: a predicting module and a blending module. The predicting module is composed of three components: a *feature learning component*, a *temporal component* with multiple ConvLSTM layers, and an *upscaling component*, as sketched in Figure 2 (a). The feature learning component extracts feature representations from the volumes, the temporal component bridges the spatial and temporal information among different volumes, while the upscaling component recovers the volumes from the spatiotemporal features. The blending module takes all outputs from the predicting module as input and generates the final synthesized volumes.

Since a traditional residual block cannot change the resolution of the input [14], we propose an advanced residual block (still referred to as the residual block for the rest of paper), which allows to downscale or upscale its input. This treatment brings several benefits to training TSR-TVD. First, it captures multiscale features. Second, it prevents the network from gradient vanishing. Third, it allows us to build a deeper neural network to enhance performance.

The core of the feature learning component is four residual blocks. Each residual block contains two parts (P_1 and P_2): P_1 consists of four Conv layers followed by spectral normalization (SN) [32] (which normalizes the parameters in the Conv layer to stabilize the training) and ReLU [33], and P_2 contains one Conv layer followed by SN and ReLU. These two parts are bridged by skip connection [41], as shown in Figure 2 (c). We set the kernel size for the first residual block to $5\times5\times5$ and for the last three residual blocks to $3\times3\times3$. We set the stride to two, which means that the resolution of the input is halved in each residual block. We set the feature maps in these four residual blocks to 16, 32, 64, and 64, respectively.

For the temporal component, we apply ConvLSTM [46] to transfer the spatial features into spatiotemporal features so that TSR-TVD can predict the next volumes via the previous volumes. Compared with the traditional LSTM [15], the weight-sharing mechanism in convolution allows us to use fewer parameters to train ConvLSTM, which saves much memory and speeds up the training process. There are three states in ConvLSTM: *input state*, *forget state*, and *output state*. These three states determine whether or not we should let new input in (input state), delete the information because the input is not important (forget state), or let the input impact the output at the current time step (output state). Taking the data ($\mathbf{x}_t$), previous hidden state ($\mathbf{h}_{t-1}$), and memory state ($\mathbf{c}_{t-1}$) as input, ConvLSTM is defined as follows

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\star\mathbf{x}_t + \mathbf{W}_{hf}\star\mathbf{h}_{t-1} + \mathbf{b}_f), \tag{10}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}\star\mathbf{x}_t + \mathbf{W}_{hi}\star\mathbf{h}_{t-1} + \mathbf{b}_i), \tag{11}$$

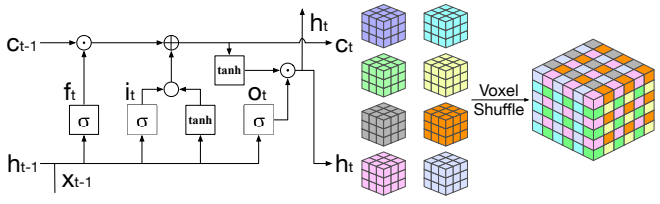
$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\star\mathbf{x}_t + \mathbf{W}_{ho}\star\mathbf{h}_{t-1} + \mathbf{b}_o), \tag{12}$$

$$\mathbf{c'}_t = \tanh(\mathbf{W}_{xc}\star\mathbf{x}_t + \mathbf{W}_{hc}\star\mathbf{h}_{t-1} + \mathbf{b}_c), \tag{13}$$

$$\mathbf{c}_t = \mathbf{i}_t\odot\mathbf{c'}_t + \mathbf{f}_t\odot\mathbf{c}_{t-1}, \tag{14}$$

$$\mathbf{h}_t = \mathbf{o}_t\odot\tanh(\mathbf{c}_t), \tag{15}$$

where $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{c}_t$, $\mathbf{o}_t$, $\mathbf{h}_t$ are the input, forget, memory, output, and hidden states at the tth step in ConvLSTM, respectively. $\sigma(\cdot)$, $\tanh(\cdot)$, $\star$, and $\odot$ represent the logistic sigmoid activation function, hyperbolic tangent activation function, Conv operation, and elementwise multiplication, respectively. $(\mathbf{W}_{xf},\mathbf{W}_{hf},\mathbf{b}_f)$, $(\mathbf{W}_{xi},\mathbf{W}_{hi},\mathbf{b}_i)$, $(\mathbf{W}_{xo},\mathbf{W}_{ho},\mathbf{b}_o)$, and $(\mathbf{W}_{xc},\mathbf{W}_{hc},\mathbf{b}_c)$ are learnable parameters in ConvLSTM. By default, we set $\mathbf{h}_0=\mathbf{0}$ and $\mathbf{c}_0=\mathbf{0}$. Note that ConvLSTM does not change the resolution of the input. An example of the ConvLSTM is shown in Figure 3 (a).



(a) (b)

Fig. 3. (a) An example of the ConvLSTM. (b) An example of the voxel shuffle layer.

The upscaling component takes the spatiotemporal features from ConvLSTM as input and outputs a synthesized volume. A common approach to recover resolution from max-pooling or Conv layers is to use deconvolutional (DeConv) layers. However, this approach brings two problems: high computational cost and unnecessary zero padding. For example, if we need to upscale a feature of size $[L,W,H]$ with a factor f, the DeConv operation will first expand the feature to a resolution of $[fL+S-1, fW+S-1, fH+S-1]$ through zero padding, where S is the kernel size, then apply Conv operations to produce the output of size $[fL, fW, fH]$. To address these problems, we propose an effective sub-voxel Conv layer, which we call the *voxel shuffle* layer, for upscaling. The definition is as follows

$$\mathbf{O} = \mathscr{S}(\mathbf{W}\star\mathbf{I}+\mathbf{b}), \tag{16}$$

where $\mathbf{I}$ and $\mathbf{O}$ are the input and output, respectively, $\mathbf{W}$ and $\mathbf{b}$ are the learnable parameters, and $\mathscr{S}$ is a periodic shuffle operation that rearranges the elements of a $[Cf^3,L,W,H]$ tensor to a tensor of size $[C,fL,fW,fH]$ (C denotes the number of channels). An example of this operation is illustrated in Figure 3 (b).

We apply the same architecture used in the feature learning component to the upscaling component. The difference is that we add the voxel shuffle layer after the last SN layer in P_1 and P_2. As for hyperparameter setting, the kernel size is set to $3\times3\times3$ in the first three residual blocks and $5\times5\times5$ in the last residual block. We set the upscaling factor to 2 for each voxel shuffle layer and the feature maps in these four residual blocks to 64, 32, 16, and 1, respectively. Note that we apply $\tanh(\cdot)$ after the final residual block.

Through the prediction module, we obtain the forward prediction $\mathbf{V}_i^{\mathrm{F}}$ and backward prediction $\mathbf{V}_i^{\mathrm{B}}$. The blending module accepts V_{k_i}, $V_{k_{i+1}}$, $\mathbf{V}_i^{\mathrm{F}}$, and $\mathbf{V}_i^{\mathrm{B}}$ as input to produce the final synthesized volume $\tilde{\mathbf{V}}_i$

$$\tilde{\mathbf{V}}_i = w_i V_{k_i} + (1-w_i)V_{k_{i+1}} + \frac{1}{2}(\mathbf{V}_i^{\mathrm{F}} + \mathbf{V}_i^{\mathrm{B}}), \tag{17}$$

where w_i is the weight which controls the importance of V_{k_i} and $V_{k_{i+1}}$.

Discriminator. To discriminate real volumes from synthesized ones, we train a discriminator network D. The architecture is shown in Figure 2 (b). Following the guidelines in Radford et al. [38] and Miyato et al. [32], we use several Conv and SN layers with leaky ReLU activation ($\alpha=0.2$), and avoid pooling layers throughout the network. D includes five Conv layers with $4\times4\times4$ kernel size, which contains 64, 128, 256, 512, and 1 feature maps, respectively. Strided convolutions are used to reduce the volume resolution at each Conv layer except the last one. The stride is set to two at the first four Conv layers. At the last Conv layer, it produces an output of size $1\times1\times1$ and we do not apply the activation function.

Table 1. The dimensions and training epochs of each data set.

data set (variable)	dimension ($x\times y\times z\times t$)	epochs
climate (cam-fv)	$360\times181\times30\times31$	100
climate (fim)	$360\times181\times30\times31$	100
combustion (HR)	$480\times720\times120\times121$	100
combustion (MF)	$480\times720\times120\times121$	100
combustion (YOH)	$480\times720\times120\times121$	100
ionization (He)	$600\times248\times248\times100$	100
ionization (He+)	$600\times248\times248\times100$	100
solar plume	$126\times126\times512\times28$	200
supernova (E)	$256\times256\times256\times60$	200
supernova (VM)	$256\times256\times256\times60$	200
vortex	$128\times128\times128\times90$	200

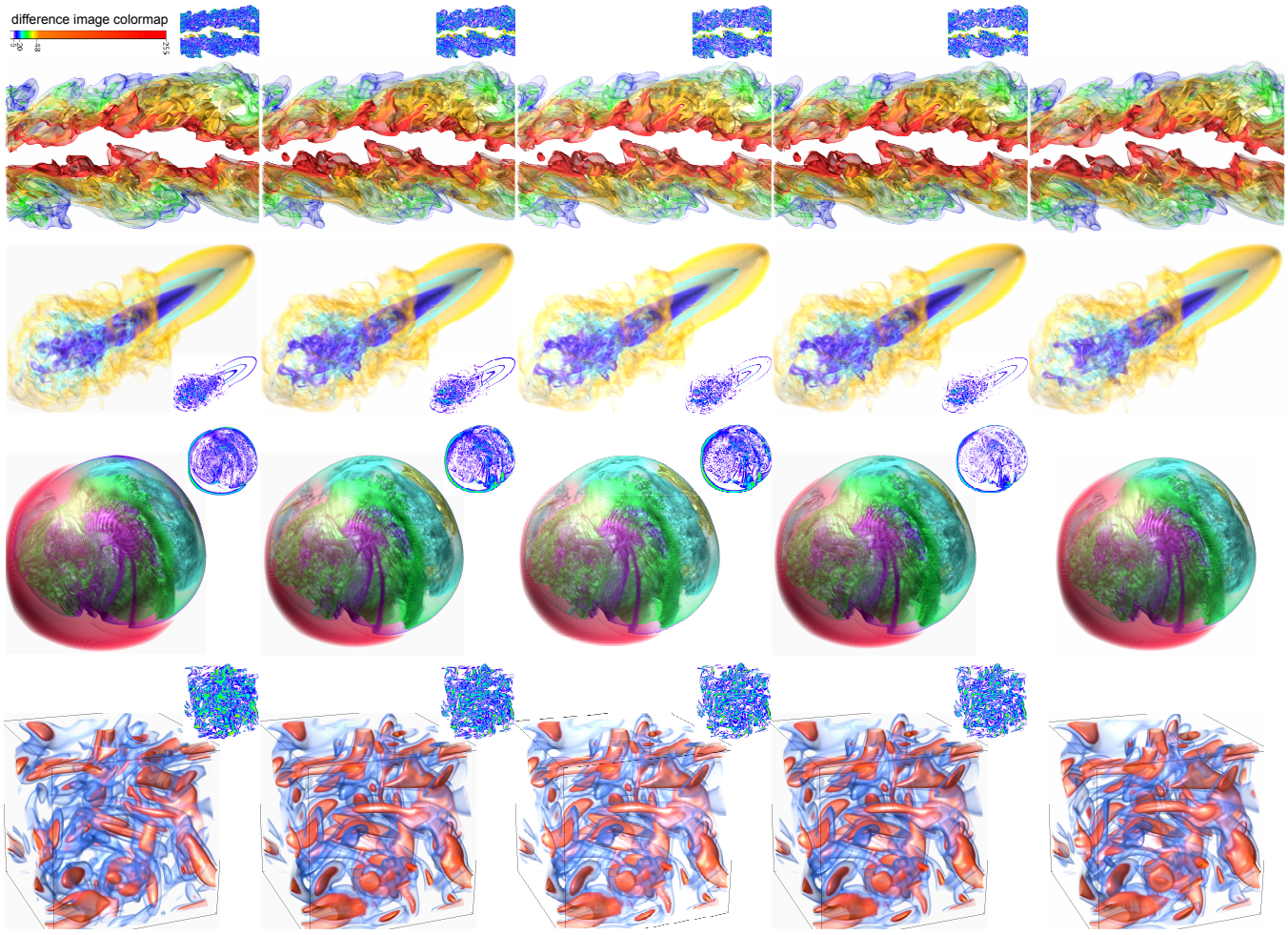
Table 2. Comparison of average PSNR and SSIM values.

	PSNR (dB)				SSIM			
data set (variable)	LERP	RNN	CNN	TSR	LERP	RNN	CNN	TSR
combustion (HR)	25.61	**26.13**	25.72	25.81	0.66	0.70	0.69	**0.72**
combustion (MF)	25.12	**25.86**	25.43	25.62	0.71	0.73	0.73	**0.74**
supernova (E)	22.34	**24.31**	23.81	23.74	0.61	0.64	0.63	**0.66**
vortex	26.62	**27.42**	26.85	26.90	0.73	0.75	0.75	**0.75**

4 Results and Discussion

4.1 Data Sets and Network Training

We experimented with TSR-TVD using the data sets listed in Table 1. The climate data set comes from the simulations of the Earth's climate in the dynamical core model intercomparison project (DCMIP) [1].



(a) LERP (b) RNN (c) CNN (d) TSR-TVD (e) GT

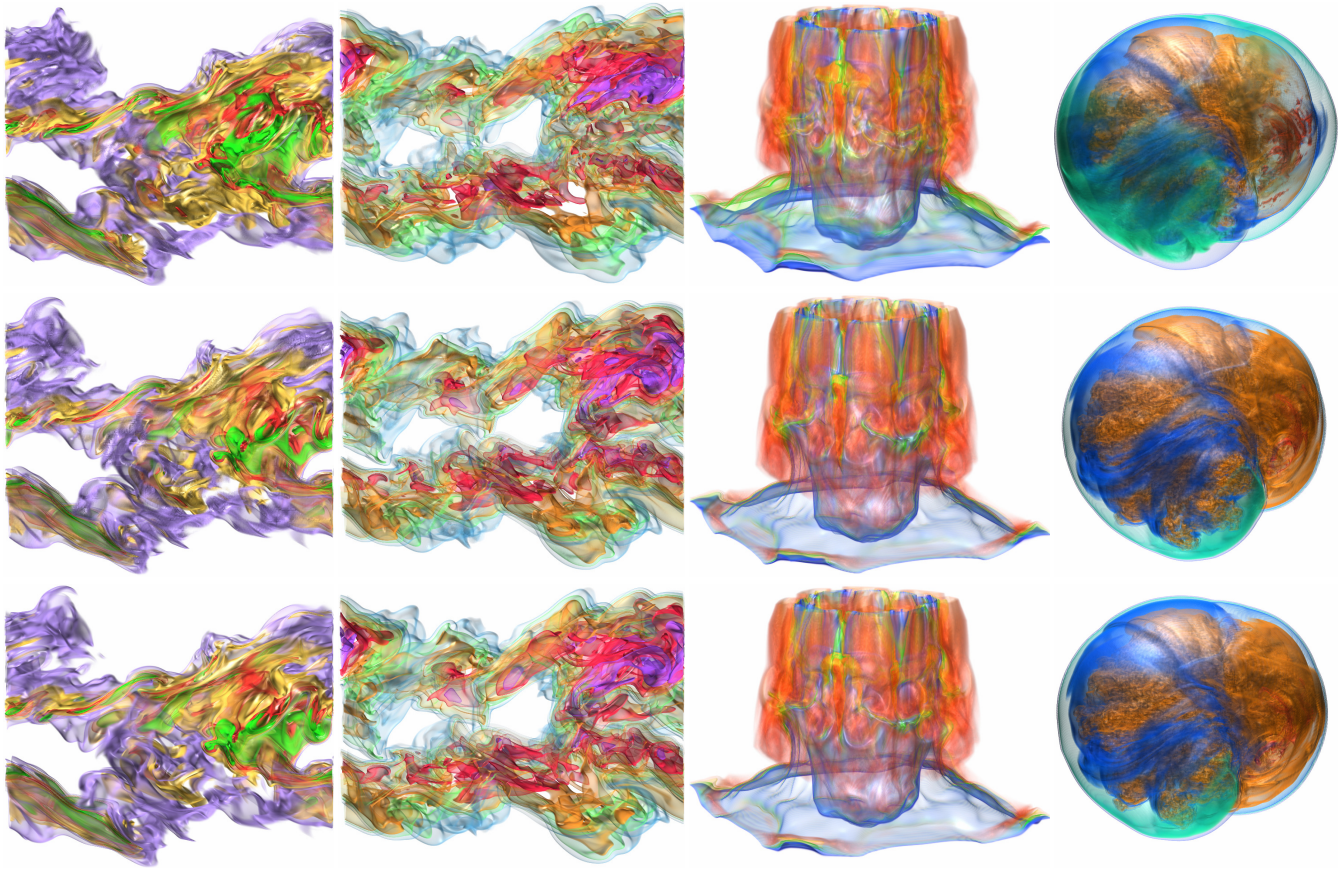
Fig. 4. Comparison of volume rendering results of same-variable inference. Top to bottom: combustion (MF), solar plume, supernova (VM), and vortex. For (a) to (d), the difference image with respect to the corresponding GT is shown at the corner.

Table 3. Comparison of average IS values at selected isovalues.

	LERP		TSR-TVD	
data set (variable)	$v = 0$	$v = 0.176$	$v = 0$	$v = 0.176$
supernova (E)	0.56	0.24	**0.71**	**0.68**
	$v = 0.255$	$v = 0.569$	$v = 0.255$	$v = 0.569$
combustion (HR)	0.65	0.59	**0.73**	**0.72**

We acquired two ensemble runs generated by different models (cam-fv and fim). In these models, the volumes are fairly static in the early time steps and later on two turbulent branches appear. The combustion data set comes from direct numerical simulation of temporally evolving turbulent non-premixed flames where combustion reactions occur within the two layers. These layers are initially thin planar layers and then evolve into complex structures as they interact with the surrounding turbulence. The simulation generates multiple variables and we used three of them: heat release (HR), stoichiometric mixture fraction (MF), and OH mass fraction (YOH). The ionization data set is made available through the IEEE Visualization 2008 Contest. The simulation is concerned with 3D radiation hydrodynamical calculations of ionization front instabilities for studying a variety of phenomena in interstellar medium such as the formation of stars. The simulation generates multiple variables and we used two of them: He mass abundance (He) and He+ mass abundance (He+). The solar plume data set comes from a simulation that aims to investigate the role of the solar plume plays in the transport of the heat, momentum, and magnetic field in the sun. The simulation generates a velocity vector field and we computed the velocity magnitude for our use. The supernova data set comes from a simulation that models the dynamics of exploding stars, which reveal instability in the shock wave blasts, imparting rotation to the newborn neutron stars in their cores. The simulation generates the entropy (E) scalar field and a velocity vector field for which we computed the velocity magnitude (VM) for our use. Finally, the vortex data set has been widely used in feature extraction and tracking. The data set comes from a pseudo-spectral simulation of vortex structures. We used the vorticity magnitude scalar variable.

A single NVIDIA TITAN Xp 1080 GPU was used for training. For each epoch, we randomly cropped four subvolumes with size $64 \times 64 \times 64$ from a volume pair $(V_{k_i}, V_{k_{i+1}})$. This cropping mechanism can speed up the training process and reduce the memory requirement. We scaled the range of inputs in $\mathbf{V}$ to $[-1,1]$ and that of output volumes to $[-1,1]$ (because the value range for the output of the final activation function $\tanh(\cdot)$ is $[-1,1]$). For optimization, we initialized parameters in TSR-TVD using those suggested by He et al. [13] and applied the Adam optimizer [26] to update the parameters. We set one training sample per minibatch. We set different learning rates for G and D in order to reduce the training time and stabilize the training process [42]. The learning rates for G and D are 10^{-4} and 4×10^{-4}, respectively. $\beta_1 = 0.0$, $\beta_2 = 0.999$. $\lambda_1 = 10^{-3}$, $\lambda_2 = 1$, and $\lambda_3 = 5 \times 10^{-2}$. We found that if λ_1 is larger than λ_2 and λ_3, TSR-TVD will fail to generate synthesized volumes that are similar to the GT volumes. This is because TSR-TVD will pay more attention to adversarial loss rather than volumetric and feature losses and the goal



(a) combustion (MF → HR) (b) combustion (MF → YOH) (c) ionization (He → He+) (d) supernova (VM → E)

Fig. 5. Comparison of volume rendering results of different-variable inference. Displayed here are combustion (HR), combustion (YOH), ionization (He+), and supernova (E). Top to bottom: LERP, TSR-TVD, and GT.

of adversarial loss is to synthesize novel volumes rather than volumes close to the GT volumes. We set ε to 10^{-4} in SN, and n_G and n_D to 1 and 2, respectively (refer to Section 1 in the Appendix). During training, we set the maximal interpolation step K to 3 because a large K would lead to gradient vanishing in ConvLSTM and prevent TSR-TVD from finding the globally optimal solution. However, during inference, we can increase K to interpolate more immediate time steps since the gradient computation is not required.

4.2 Results

Due to the page limit, we are not able to show TSR-TVD results for multiple time steps in the paper. These results can be found in the accompanying video, which shows visualization results over the entire sequence and highlights the better quality of synthesized volumes generated using TSR-TVD over LERP, RNN, and CNN. Unless otherwise stated, all visualization results presented in the paper for volumes synthesized by TSR-TVD are the inferred results (i.e., the network does not see these volumes during training). These influence results are from a volume subsequence that is far away from for the training data, and within the subsequence, we select the time step that is farthest away from the GT volumes at the two end time steps (i.e., we show the worst possible TSR-TVD results). All visualizations for the same data set use the same setting for lighting, viewing, and transfer function (for volume rendering). For network analysis, please refer to Section 2 in the Appendix.

Evaluation metrics. We utilize the *peak signal-to-noise ratio* (PSNR) to evaluate the quality of synthesized volumes at the data level. PSNR is defined as

$$\mathrm{PSNR}(\mathbf{V},\mathbf{V}') = 20\log_{10} I(\mathbf{V}) - 10\log_{10}\mathrm{MSE}(\mathbf{V},\mathbf{V}'), \tag{18}$$

where $\mathbf{V}$ and $\mathbf{V}'$ are the original and synthesized volumes, $I(\mathbf{V})$ is the difference between the maximum and minimum values of $\mathbf{V}$, and $\mathrm{MSE}(\mathbf{V},\mathbf{V}')$ is the mean squared error between $\mathbf{V}$ and $\mathbf{V}'$.

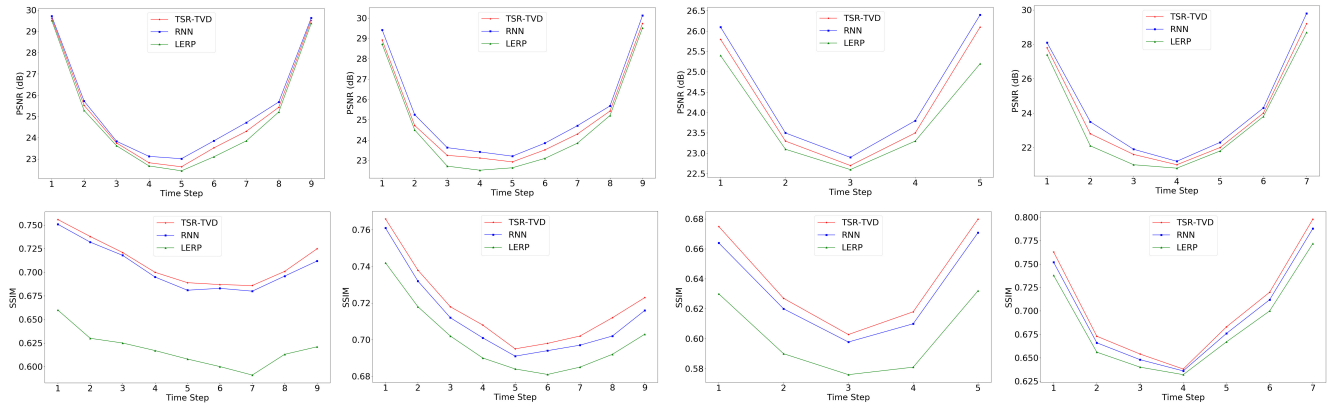
We apply the *structural similarity index* (SSIM) [56] to evaluate the quality of rendered images at the image level. SSIM is defined as

$$\mathrm{SSIM}(\mathbf{I},\mathbf{I}') = \frac{(2\mu_{\mathbf{I}}\mu_{\mathbf{I}'}+c_1)(2\sigma_{\mathbf{I},\mathbf{I}'}+c_2)}{(\mu_{\mathbf{I}}^2+\mu_{\mathbf{I}'}^2+c_1)(\sigma_{\mathbf{I}}^2+\sigma_{\mathbf{I}'}^2+c_2)}, \tag{19}$$

where $\mathbf{I}$ and $\mathbf{I}'$ are sub-images from the rendered images of $\mathbf{V}$ and $\mathbf{V}'$, $\mu_{\mathbf{I}}$ and $\mu_{\mathbf{I}'}$ are the average values of $\mathbf{I}$ and $\mathbf{I}'$, $\sigma_{\mathbf{I}}^2$ and $\sigma_{\mathbf{I}'}^2$ are the variances of $\mathbf{I}$ and $\mathbf{I}'$, $\sigma_{\mathbf{I},\mathbf{I}'}$ is the covariance of $\mathbf{I}$ and $\mathbf{I}'$, and c_1 and c_2 are two small constants for stabilizing the division with the denominator.

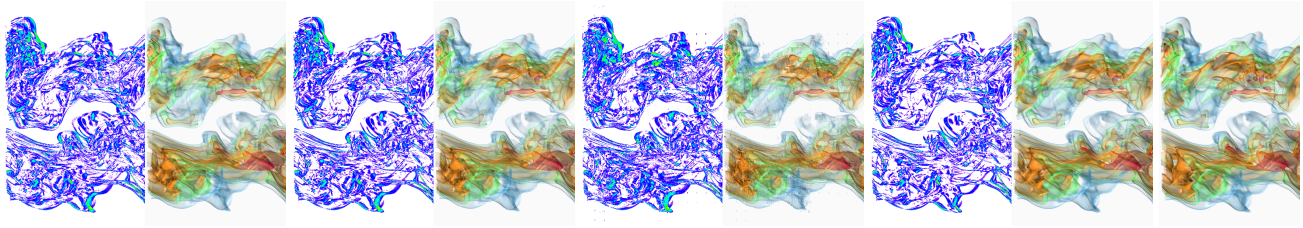
To quantify the similarity between two isosurfaces extracted, respectively, from the synthesized and GT volumes, we compute the mutual information of their corresponding distance fields [5]. Mutual information is computed by constructing a joint histogram of the two distance fields, where each entry (i,j) in the joint histogram contains the number of voxels that fall into bins i and j in the first and second distance fields, respectively. The larger the *isosurface similarity* (IS), the more similar the two surfaces are.

Qualitative and quantitative analysis. In Figure 4, we compare the rendering results of the synthesized volumes generated by LERP, RNN, and TSR-TVD. To train an RNN, we still use the architecture of TSR-TVD but eliminate the discriminator. To train a CNN, we use the same TSR-TVD architecture without ConvLSTM as the built-in temporal coherence predictor. For easy comparison, we calculate pixelwise differences (the Euclidean distances) of images generated from the synthesized and original volumes in the CIELUV color space. We map the noticeable pixel differences (with $\Delta \geqslant 6.0$) to nonwhite colors (clamping differences larger than 255) and display the difference image at the corner. The GT is provided for a fair comparison. For the combustion (MF) data set, RNN, CNN, and TSR-TVD can generate



(a) combustion (HR) (b) combustion (MF) (c) supernova (E) (d) vortex

Fig. 6. Comparison of PSNR of synthesized volumes (top row) and SSIM of rendered images (bottom row) using LERP, RNN, and TSR-TVD. The projection views for computing SSIM are shown in the respective volume-rendering images in the paper. We only report PSNR and SSIM values for a volume subsequence where the GT time steps are available at both ends. The results for other subsequences follow a similar trend.



(a) w/o fwd prediction (b) w/o bwd prediction (c) w/o original volumes (d) full model (e) GT

Fig. 7. Comparison of volume rendering results of same variable inference with the variants of TSR-TVD using the combustion (YOH) data set. Displayed here are the cropped left-half of the rendering images.

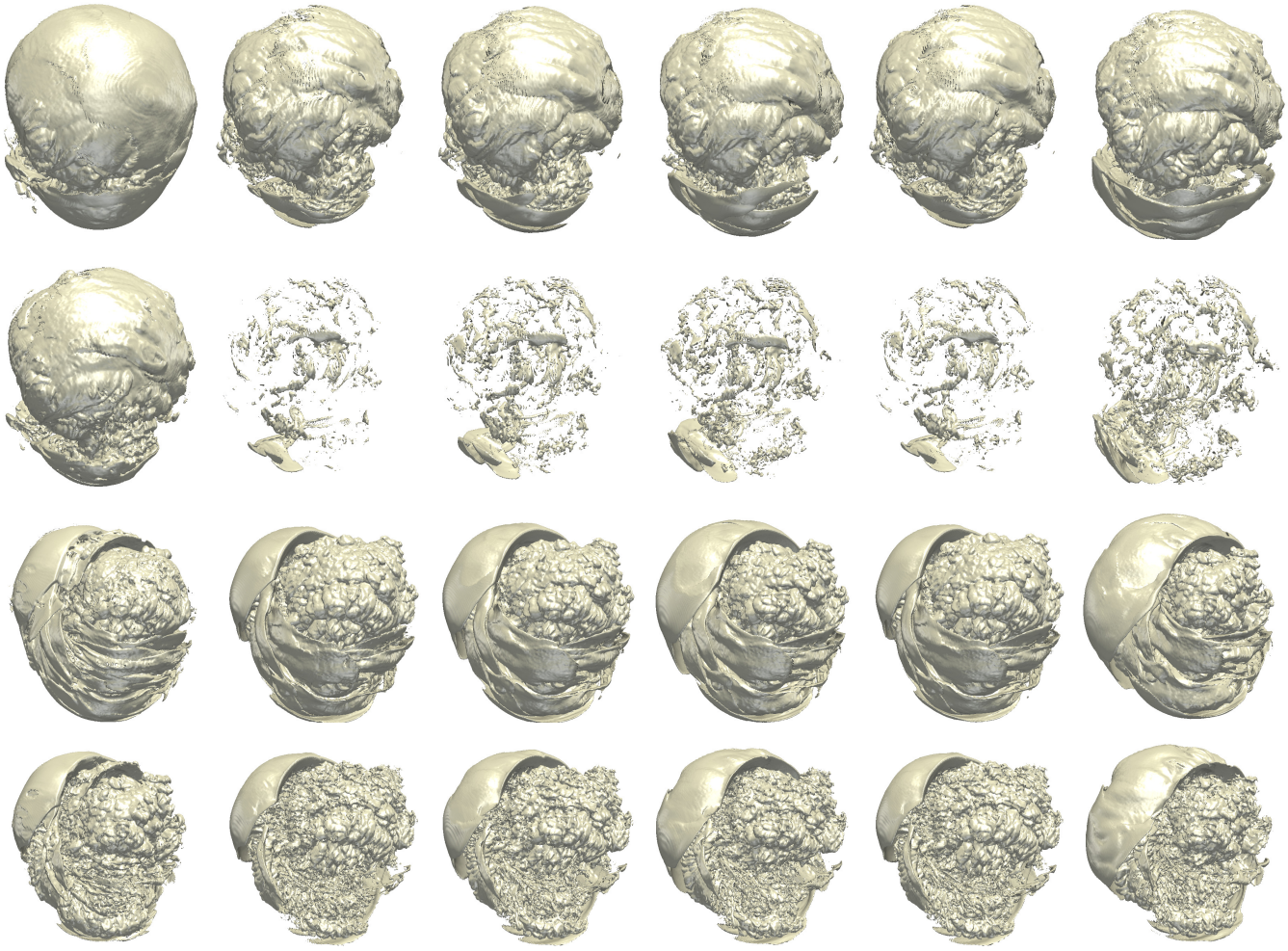
more details compared with LERP, for example, the small red component at the left corner. But the rendering result from TSR-TVD has fewer artifacts than the result from RNN or CNN. For the solar plume data set, all four methods generate similar results, but TSR-TVD is closest to the GT. It is clear that TSR-TVD leads to the best visual quality for the supernova (VM) data set while LERP is the worst and RNN or CNNgenerates many artifacts on the right side. For the vortex data set, LERP generates the worst result while RNN, CNN, and TSR-TVD yield similar results. Upon close examination, we can see that TSR-TVD yields the closest result at the bottom-left corner.

In Figure 5, we compare the rendering results of the synthesized volumes generated by LERP and TSR-TVD through different-variable inference. For different-variable inference, we use a variable X of a data set for training and use another variable Y of the same data set for inference ($X \rightarrow Y$). For MF $\rightarrow$ HR of the combustion data set, TSR-TVD produces high-quality and detailed visual results in the purple part at the top-left corner and in the green and yellow parts at the middle-right corner while LERP yields less accurate results. For MF $\rightarrow$ YOH of the combustion data set, TSR-TVD generates more accurate rendering result compared with LERP. For example, the purple part at the top-right corner and the green part at the bottom-right corner are closer to the GT. As for He $\rightarrow$ He+ of the ionization data set, it is clear that TSR-TVD gives high-quality visual results at the middle and bottom layers while LERP leads to color shifting due to content change (even though the same transfer function is used to render the GT and synthesized volumes). It is obvious that TSR-TVD generates a better visual result for VM $\rightarrow$ E of the supernova data set. TSR-TVD produces more accurate details in the orange, navy blue, and cyan parts. LERP, however, cannot faithfully recover these parts as we can clearly see color shifting due to content change.

In Figure 6, we compare TSR-TVD results against LERP and RNN results at the data (PSNR) and image (SSIM) levels. At the data level, we can see that RNN achieves the highest PSNR values. This is because RNN is a PSNR-oriented solution while TSR-TVD is constrained by not only the volumetric loss (PSNR-oriented loss) but also adversarial (perception-oriented loss) and feature losses, which could lead to lower PSNR values. For these four data sets, the PSNR curves follow a similar trend: PSNR values peak at both ends of the volume subsequence where the GT time steps are available and fall steadily as we move toward the time steps in the middle of the interval. We also see lower PSNR values across the three methods for the supernova (E) data set. This is because the supernova exhibits a fast-pacing rotational behavior which is more difficult to capture compared with other behaviors exhibited by the other two data sets. At the image level, TSR-TVD yields the highest SSIM values. It is a clear winner for the combustion (HR), combustion (MF), and supernova (E) data sets. TSR-TVD produces average SSIM values of ~ 0.72, ~ 0.72, ~ 0.65, respectively, but LERP produces average SSIM values of ~ 0.62, ~ 0.70, ~ 0.60, respectively. For the vortex data set, TSR-TVD still slightly outperforms LERP and RNN. In Table 2, we report the average PSNR and SSIM values over the entire volume sequence for LERP, RNN, CNN, and TSR-TVD. Again, RNN performs the best in terms of PSNR while TSR-TVD performs the best in terms of SSIM.

In Figure 7, we show the variants of TSR-TVD using the combustion (YOH) data set. We can see that only using forward or backward prediction can still generate the synthesized volumes reasonably well but the results lack fine details. For example, there is a closed light cyan part in the GT volume at the middle-right corner, which is captured in the full mode in Figure 7 (d) but missed in both Figure 7 (a) and (b). Moreover, we find that without adding the original volumes into the blending module, the rendering result leads to obvious noise, as shown in Figure 7 (c). This is because we use a traditional down-sampling and up-sampling framework to reconstruct intermediate volumes. In the down-sampling phase, TSR-TVD will compress the volumes and lose some information, however, in the up-sampling phase, the lost information cannot be perfectly recovered. This kind of information loss leads to the inferior quality of rendering images.

In Figures 8 and 9, we compare the isosurface rendering results of the synthesized volumes generated by LERP, TSR-TVD without blending original volumes, and TSR-TVD using the supernova (E) and



(a) LERP (b) w/o ori volumes (c) TSR-TVD (d) GT (e) left-end GT (f) right-end GT

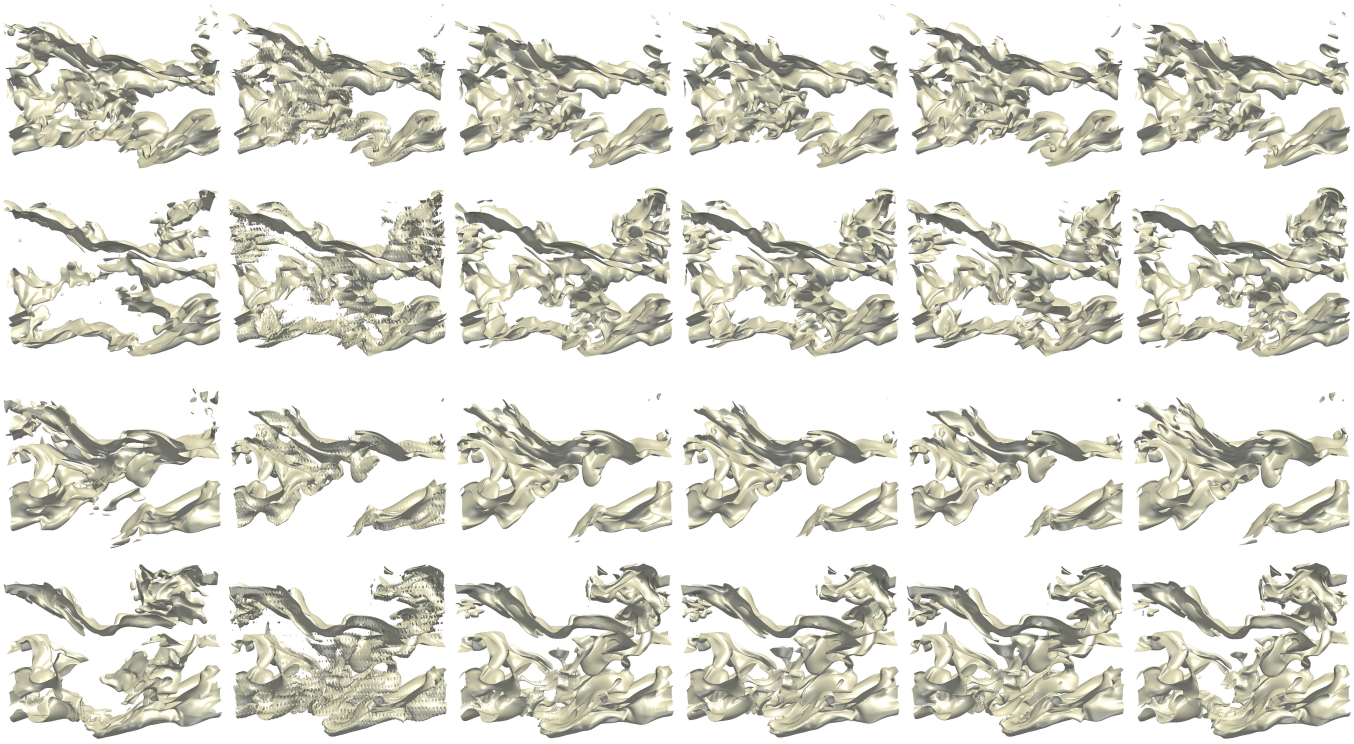
Fig. 8. Comparison of isosurface rendering results of same-variable inference using the supernova (E) data set. First row: time steps 39 with $v = 0$. Second row: time steps 39 with $v = 0.176$. Third row: time steps 55 with $v = 0$. Last row: for time steps 55 with $v = 0.176$. The two-end GT time steps are 37 and 41 for time step 39, and 53 and 57 for time step 55.

combustion (HR) data sets. For each data set, the value range is normalized to $[-1, 1]$ and we pick two time steps and two isovalues to show the isosurfaces. For the supernova (E) data set, we can observe that for $v = 0$, the isosurface generated by TSR-TVD includes more details (e.g., the bottom-right corner for time step 39 and the top-right corner for time step 55). For $v = 0.176$, TSR-TVD produces a higher-quality isosurface since LERP totally fails to construct the isosurface close to the GT at time step 39 (we can see that the surface is severely shifted in the value space). Adding backward and forward predictions along with the use of voxel-wise volumetric loss, TSR-TVD is able to largely mitigate this and produces a surface very close to the GT. For the combustion (HR) data set, TSR-TVD can still generate closer isosurfaces than LERP, such as the bottom-left corner ($v = 0.255$) and the top-right corner ($v = 0.569$). For both data sets, it is clear that the full mode of TSR-TVD generates better results than those without adding the original volumes into the blending module. In addition, we report in Table 3 the average IS values over the entire volume sequence for these two data sets. The quantitative results also confirm that TSR-TVD leads to isosurfaces of better quality than LERP.

Failure case. As shown in Figure 10, both LERP and TSR-TVD cannot interpolate the intermediate time steps well for the climate (fim) data set. We can see that voxel values are shifted in both results. For example, the green parts in the GT become yellow and the blue parts almost vanish in the results of LERP and TSR-TVD. This is due to the limitation of TSR-TVD in estimating the difference between data distributions of neighboring time steps. Refer to Section 3 in the Appendix for further discussion.

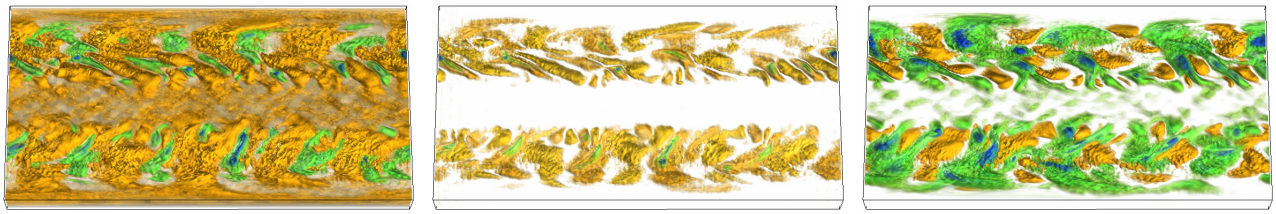
4.3 Discussion

As a deep learning technique, TSR-TVD takes a considerable amount of time for training. In Figure 2 in the Appendix, we report the training time curves for selected data sets under different hyperparameter settings. With 40% of training samples and subvolume size of $64 \times 64 \times 64$, it takes nearly six hours to train the combustion data set (100 epochs) and half a day to train the supernova data set (200 epochs). The training time forms a nearly linear relationship with the increase in the number of training samples or the subvolume size. We note that the actual size of the volume (i.e., the spatial dimension of the data) is not a critical limiting factor for network training because we apply the cropping mechanism in TSR-TVD training. Therefore, the training time mainly depends on the (cropped) subvolume size rather than the volume size. The inference time depends on the number of interpolated time steps and volume resolution. Due to the limited GPU memory, we infer individual subvolumes to form the entire volume. The time is between one hour (vortex) to one day (combustion). Refer to Section 3 in the Appendix for further discussion. Our current TSR-TVD framework has the following limitations. First, our solution does not consider the input transfer function or the visualization process. We simply treat the input 3D volumes as numerical data for producing the temporally resolved results. In terms of visualization, we use 1D transfer functions that map scalar values to color and opac-



(a) LERP (b) w/o ori volumes (c) TSR-TVD (d) GT (e) left-end GT (f) right-end GT

Fig. 9. Comparison of isosurface rendering results of same-variable inference using the combustion (HR) data set. First row: time steps 97 with $v = 0.255$. Second row: time steps 97 with $v = 0.569$. Third row: time steps 114 with $v = 0.255$. Last row: for time steps 114 with $v = 0.569$. The two-end GT time steps are 95 and 99 for time step 97, and 112 and 116 for time step 114.



(a) LERP (b) TSR-TVD (c) GT

Fig. 10. Comparison of volume rendering results of same-variable inference using the climate (fim) data set.

ity. The volume renderer also implements lighting calculation. Nevertheless, qualitative and quantitative evaluations show the overall advantage of TSR-TVD over LERP and RNN. While incorporating the transfer function, especially the opacity transfer function, may help boost the performances at the image level, it comes at the price of making the training process dependent on the input transfer function, which would demand the training from scratch whenever the transfer function changes. Second, TSR-TVD can only infer the intermediate volumes at any integer time steps rather than arbitrary time steps. We would further study how to infer arbitrary in-between volumes from two given volumes through disentangled representation [8, 17]. Third, TSR-TVD considers temporal coherence through the recurrent module. A better way is to incorporate temporal coherence into loss function design. We will investigate temporal and cycle losses that take the coherence of neighboring time steps into consideration to achieve better temporal coherence.

5 Conclusions and Future Work

We have presented TSR-TVD, a deep learning solution for generating temporal super-resolution of time-varying data. Given a volume pair as input and its intermediate volumes as the GT, we leverage an RGN to make forward and backward predictions and train TSR-TVD. The trained network is then able to generate temporally resolved volume sequences for the rest of time series via same-variable inference or different-variable inference. Compared with LERP, RNN, and CNN, TSR-TVD yields synthesized intermediate volumes of better visual quality. TSR-TVD is part of our research effort toward what we call *data augmentation for scientific visualization*. Data augmentation in this context refers to the addition of spatial, temporal, and variable details to reduced data by incorporating information derived from internal and external sources. Besides temporal super-resolution (TSR), we would also consider spatial super-resolution (SSR) for time-varying data. Our eventual goal is to achieve spatiotemporal super-resolution (STSR) by producing volume sequences with greater spatial and temporal resolutions and details. The ability to upscale time-varying data in both spatial and temporal dimensions is critical for large-scale scientific simulations and applications. As scientists often have to save their simulation data sparsely due to the limited storage, our research will provide a promising alternative for them to make better decisions depending on the nature of the simulations and the characteristics of the data.

Acknowledgments

This research was supported in part by the U.S. National Science Foundation through grants IIS-1455886, CNS-1629914, and DUE-1833129, and the NVIDIA GPU Grant Program. The authors would like to thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] The 2012 dynamical core model intercomparison project (DCMIP). https://earthsystemcog.org/projects/dcmip-2012/.

[2] H. Akiba and K.-L. Ma. A tri-space visualization interface for analyzing time-varying multivariate volume data. In *Proceedings of Eurographics - IEEE VGTC Symposium on Visualization*, pages 115–122, 2007.

[3] M. Berger, J. Li, and J. A. Levine. A generative model for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1636–1650, 2019.

[4] A. Biswas, S. Dutta, H.-W. Shen, and J. Woodring. An information-aware framework for exploring multivariate data sets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2683–2692, 2013.

[5] S. Bruckner and T. Möller. Isosurface similarity maps. *Computer Graphics Forum*, 29(3):773–782, 2010.

[6] H. Cai, C. Bai, Y.-W. Tai, and C.-K. Tang. Deep video generation, prediction and completion of human action sequences. In *Proceedings of European Conference on Computer Vision*, pages 374–390, 2018.

[7] H.-C. Cheng, A. Cardone, S. Jain, E. Krokos, K. Narayan, S. Subramaniam, and A. Varshney. Deep-learning-assisted volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 25(2):1378–1391, 2019.

[8] E. L. Denton and V. Birodkar. Unsupervised learning of disentangled representations from video. In *Proceedings of Advances in Neural Information Processing Systems*, pages 4414–4423, 2017.

[9] M. Glatter, J. Huang, S. Ahern, J. Daniel, and A. Lu. Visualizing temporal patterns in large multivariate data using textual pattern matching. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1467–1474, 2008.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[11] 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*, 2019. Accepted.

[12] J. Han, J. Tao, H. Zheng, H. Guo, D. Z. Chen, and C. Wang. Flow field reduction via reconstructing vector data from 3D streamlines using deep learning. *IEEE Computer Graphics and Applications*, 39(4):54–67, 2019.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[16] F. Hong, J. Zhang, and X. Yuan. Access pattern learning with long short-term memory for parallel particle tracing. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 76–85, 2018.

[17] J.-T. Hsieh, B. Liu, D.-A. Huang, F.-F. Li, and J. C. Niebles. Learning to decompose and disentangle representations for video prediction. In *Proceedings of Advances in Neural Information Processing Systems*, pages 515–524, 2018.

[18] Z. Hu, Y. Ma, and L. Ma. Multi-scale video frame-synthesis network with transitive consistency loss. *arXiv preprint arXiv:1712.02874*, 2017.

[19] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134, 2017.

[20] T. J. Jankun-Kelly and K.-L. Ma. A study of transfer function generation for time-varying volume data. In *Proceedings of Eurographics - IEEE TCVG Workshop on Volume Graphics*, pages 51–66, 2001.

[21] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super SloMo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[22] J. Johnson, A. Alahi, and F.-F. Li. Perceptual losses for real-time style transfer and super-resolution. In *Proceedings of European Conference on Computer Vision*, pages 694–711, 2016.

[23] A. Joshi and P. Rheingans. Illustration-inspired techniques for visualizing time-varying data. In *Proceedings of IEEE Visualization Conference*, pages 679–686, 2005.

[24] J. Kehrer and H. Hauser. Visualization and visual analysis of multifaceted scientific data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):495–513, 2013.

[25] B. Kim and T. Günther. Robust reference frame extraction from unsteady 2D vector fields with convolutional neural networks. *Computer Graphics Forum*, 38(3), 2019.

[26] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of International Conference for Learning Representations*, 2015.

[27] W.-S. Lai, J.-B. Huang, O. Wang, E. Shechtman, E. Yumer, and M.-H. Yang. Learning blind video temporal consistency. In *Proceedings of European Conference on Computer Vision*, pages 170–185, 2018.

[28] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017.

[29] X. Liu and H.-W. Shen. Association analysis for visual exploration of multivariate scientific data sets. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):955–964, 2016.

[30] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of IEEE International Conference on Computer Vision*, pages 4473–4481, 2017.

[31] K.-L. Ma. Machine learning to boost the next generation of visualization technology. *IEEE Computer Graphics and Applications*, 27(5):6–9, 2007.

[32] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. In *Proceedings of International Conference for Learning Representations*, 2018.

[33] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of International Conference on Machine Learning*, pages 807–814, 2010.

[34] A.-D. Nguyen, W. Kim, J. Kim, and S. Lee. Video frame interpolation by plug-and-play deep locally linear embedding. *arXiv preprint arXiv:1807.01462*, 2018.

[35] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–679, 2017.

[36] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of IEEE International Conference on Machine Learning*, pages 1310–1318, 2013.

[37] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[38] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[39] M. Raji, A. Hota, R. Sisneros, P. Messmer, and J. Huang. Photo-guided exploration of volume data features. In *Proceedings of Eurographics Symposium on Parallel Graphics and Visualization*, pages 31–39, 2017.

[40] A. Ramdas, N. Trillos, and M. Cuturi. On Wasserstein two-sample testing and related families of nonparametric tests. *Entropy*, 19(2):47–62, 2017.

[41] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241, 2015.

[42] K. Roth, A. Lucchi, S. Nowozin, and T. Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Proceedings of Advances in Neural Information Processing Systems*, pages 2018–2028, 2017.

[43] N. Sauber, H. Theisel, and H.-P. Seidel. Multifield-Graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):917–924, 2006.

[44] H.-W. Shen, L.-J. Chiang, and K.-L. Ma. A fast volume rendering algorithm for time-varying fields using a time-space partitioning (TSP) tree. In *Proceedings of IEEE Visualization Conference*, pages 371–377, 1999.

[45] N. Shi and Y. Tao. CNNs based viewpoint estimation for volume visualization. *ACM Transactions on Intelligent Systems and Technology*, 10(3):27:1–27:22, 2019.

[46] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo. Convolutional LSTM network: A machine learning approach for precip-

itation nowcasting. In *Proceedings of Advances in Neural Information Processing Systems*, pages 802–810, 2015.

[47] K. Stockinger, J. Shalf, K. Wu, and E. W. Bethel. Query-driven visualization of large data sets. In *Proceedings of IEEE Visualization Conference*, pages 167–174, 2005.

[48] J. Tao, M. Imre, C. Wang, N. V. Chawla, H. Guo, G. Sever, and S. H. Kim. Exploring time-varying multivariate volume data using matrix of isosurface similarity maps. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1236–1245, 2019.

[49] F.-Y. Tzeng, E. B. Lum, and K.-L. Ma. A novel interface for higher-dimensional classification of volume data. In *Proceedings of IEEE Visualization Conference*, pages 505–512, 2003.

[50] F.-Y. Tzeng, E. B. Lum, and K.-L. Ma. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):273–284, 2005.

[51] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *Proceedings of IEEE International Conference on Computer Vision*, pages 4534–4542, 2015.

[52] C. Wang and H.-W. Shen. A framework for rendering large time-varying data using wavelet-based time-space partitioning (WTSP) tree. Technical Report OSU-CISRC-1/04-TR05, Department of Computer and Information Science, The Ohio State University, 2004.

[53] C. Wang, H. Yu, R. W. Grout, K.-L. Ma, and J. H. Chen. Analyzing information transfer in time-varying multivariate data. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 99–106, 2011.

[54] C. Wang, H. Yu, and K.-L. Ma. Importance-driven time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1547–1554, 2008.

[55] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[56] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[57] S. Weiss, M. Chu, N. Thuerey, and R. Westermann. Volumetric isosurface rendering with deep learning-based super-resolution. *arXiv preprint arXiv:1906.06520*, 2019.

[58] S. Wiewel, M. Becher, and N. Thuerey. Latent-space physics: Towards learning the temporal evolution of fluid flow. *Computer Graphics Forum*, 38(2):71–82, 2019.

[59] J. Woodring, C. Wang, and H.-W. Shen. High dimensional direct rendering of time-varying volumetric data. In *Proceedings of IEEE Visualization Conference*, pages 417–424, 2003.

[60] Y. Xie, E. Franz, M. Chu, and N. Thuerey. tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Transactions on Graphics*, 37(4):95:1–95:15, 2018.

[61] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[62] Z. Zhou, Y. Hou, Q. Wang, G. Chen, J. Lu, Y. Tao, and H. Lin. Volume upscaling with convolutional neural networks. In *Proceedings of Computer Graphics International*, pages 38:1–38:6, 2017.

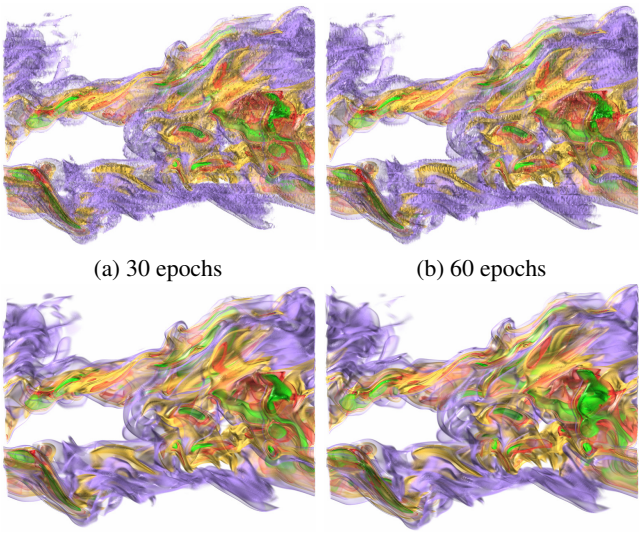
APPENDIX

1 TRAINING ALGORITHM AND OPTIMIZATION

Algorithm. As sketched in Algorithm 1, our TSR-TVD training algorithm contains two parts: optimizing D (discriminator) and G (generator). The algorithm runs over a certain number of epochs T and for each epoch, it optimizes the two networks with n_D and n_G times, respectively. In order to generate high-quality results, the discriminator must be powerful, which means that D should distinguish the synthesized results as fake ones even if they only have few differences. Therefore, we usually set $n_D \geqslant n_G$.

Algorithm 1 TSR-TVD training algorithm.

Require: initial generator parameters θ_G and initial discriminator parameters θ_D
Require: number of D updates n_D per G iteration, number of G updates n_G per D iteration, number of training epochs T, learning rates α_G and α_D for G and D, respectively
for $t = 1 \cdots T$ **do**
  for $i = 1 \cdots n$ **do**
    sample $(V_{k_i}, V_{k_{i+1}})$ and $(V_{k_i+1}, \cdots, V_{k_{i+1}-1})$
    for $1 \cdots n_D$ **do**
      compute $\mathscr{L}_D$ according to Equation 6
      $\theta_D = \theta_D - \alpha_D \frac{\partial \mathscr{L}_D}{\partial \theta_D}$
    end for
    for $1 \cdots n_G$ **do**
      compute $\mathscr{L}_G$ according to Equation 9
      update θ_G according to Equation 21
    end for
  end for
end for



(a) 30 epochs (b) 60 epochs (c) 100 epochs (d) GT

Fig. 1. Comparison of volume rendering results under different training epochs using the combustion (HR) data set. The best match with the GT is the result with 100 epochs.

Optimization. We apply *backpropagation through time* (BPTT), a gradient-based approach to optimize TSR-TVD. Compared with traditional optimization techniques, such as evolutionary optimization, BPTT can accelerate the training process. We formulate BPTT from the ith ConvLSTM (L^i) to the $(i+j)$th ConvLSTM (L^{i+j}) as follows

$$\frac{\partial L^{i+j}}{\partial L^i} = \prod_{t=i+1}^{i+j} \frac{\partial L^t}{\partial L^{t-1}}. \tag{20}$$

The detail for computing $\frac{\partial L^t}{\partial L^{t-1}}$ is in Hochreiter and Schmidhuber [15].

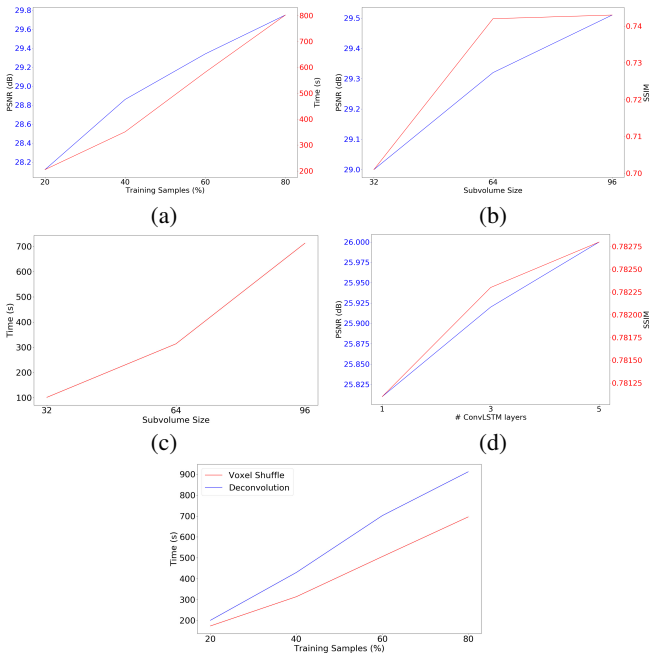


Fig. 2. Comparison of hyperparameter settings. (a) Average PSNR and training time (per epoch) under different training samples using the ionization (He) data set. (b) Average PSNR and SSIM under different subvolume sizes using the combustion (MF) data set. (c) Average training time (per epoch) under different subvolume sizes using the combustion (MF) data set. (d) Average PSNR and SSIM under different ConvLSTM layers using the combustion (HR) data set. (e) Average training time (per epoch) under different upscaling methods using the vortex data set.

We then update the generator parameter θ_G through

$$\theta_G = \theta_G - \alpha_G \sum_{i=1}^{K} \frac{\partial \mathscr{L}_G}{\partial U^i} \frac{\partial U^i}{\partial L^K} \frac{\partial L^K}{\partial L^i} \frac{\partial L^i}{\partial F^i} \frac{\partial F^i}{\partial \theta_G}, \tag{21}$$

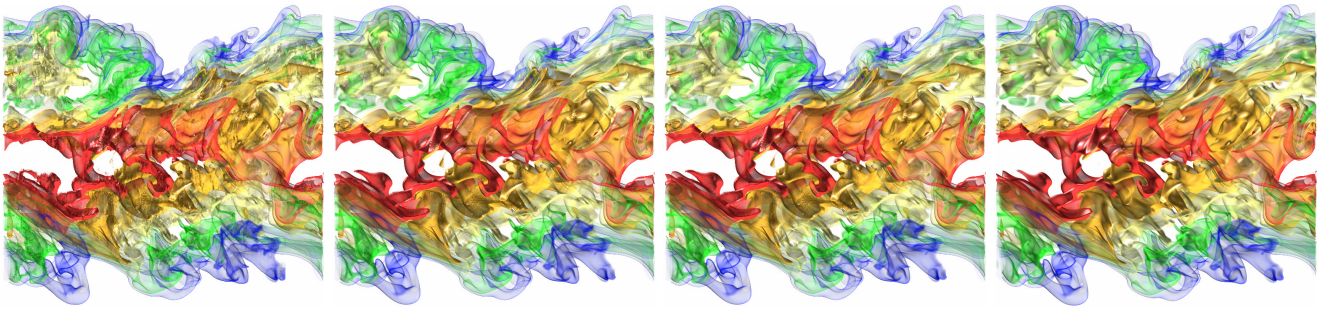
where F^i and U^i are the ith feature learning and upscaling components, K is the maximal interpolation step, and α_G is the learning rate of G.

2 NETWORK ANALYSIS

In order to evaluate TSR-TVD, we analyze the following hyperparameter settings: training epochs, training samples, subvolume size, number of ConvLSTM layers, maximal interpolation step, and upscaling methods. A detailed discussion is as follows.

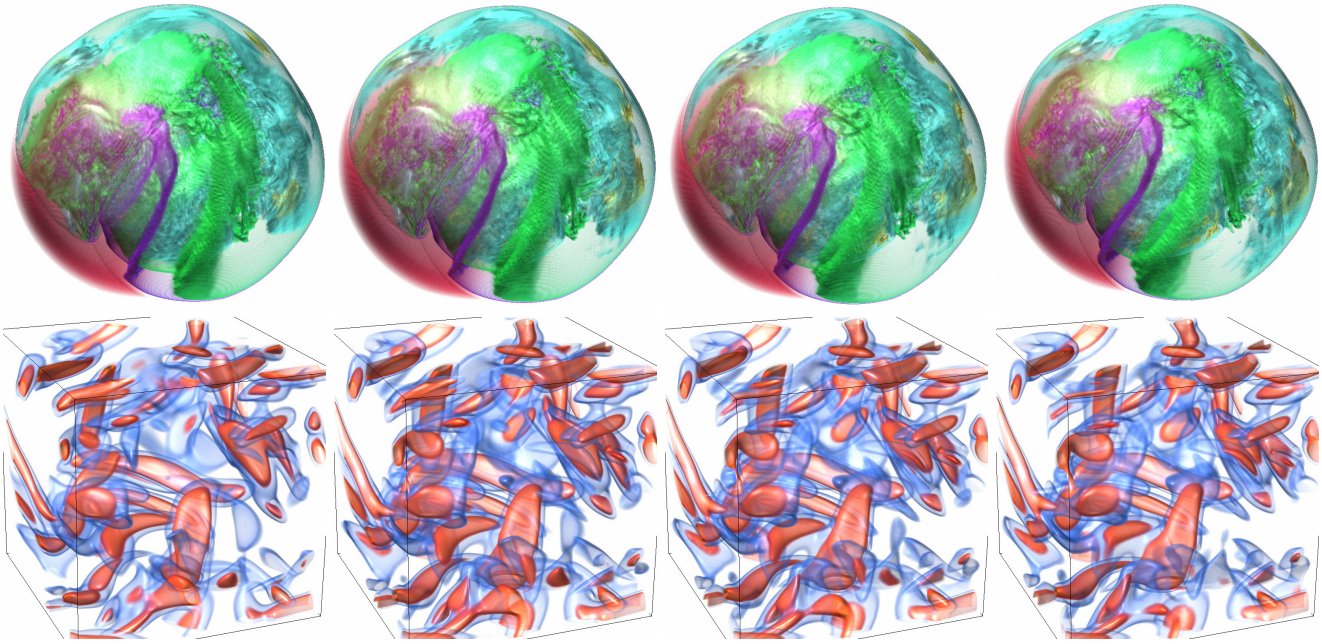
Training epochs vs. visual quality. We investigate how the visual quality of the synthesized volume using TSR-TVD evolves with different training epochs with the combustion (HR) data set. Rendering of the volumes obtained after different numbers of training epochs is illustrated in Figure 1. We can observe that the rendered images under 30 and 60 epochs have more blurring artifacts compared with the result under 100 epochs. For example, the purple part at the top-left corner and the green part at the middle-right corner have fewer artifacts as the training goes. Moreover, we find that beyond 100 epochs, there is no significant difference among synthesized results. Therefore, we choose 100 epochs to train the combustion data set. The same experiment is taken to decide the training epochs for other data sets.

Training samples vs. PSNR. We study the influence of the number of training samples on PSNR and training time. For example, 20% training samples for a time-varying data set of 100 time steps mean that we have the first 20 time steps available as the GT data during training. We use 20%, 40%, 60%, and 80% training samples to train TSR-TVD using the ionization (He) data set. We plot the PSNR and average training time curves under different numbers of training samples, as shown in Figure 2 (a). We can see that using more training samples



(a) $32 \times 32 \times 32$ (b) $64 \times 64 \times 64$ (c) $96 \times 96 \times 96$ (d) GT

Fig. 3. Comparison of volume rendering results under different subvolume sizes using the combustion (MF) data set.



(a) (b) (c) (d)

Fig. 4. Comparison of volume rendering results under different maximal interpolation steps. Top row: supernova (VM) data set. Bottom row: vortex data set. (a) shows the GT. (b) to (d) show the rendering results with $K = 3, 7, 11$ (supernova) and $K = 3, 5, 7$ (vortex), respectively.

improves PSNR. However, this demands a longer training time. Moreover, we observe that visual quality does not benefit significantly from using more training samples. To achieve a balance between quality and speed, we use 40% samples to train TSR-TVD.

Subvolume size vs. visual quality and PSNR. We train TSR-TVD with subvolume sizes of $32 \times 32 \times 32$, $64 \times 64 \times 64$, and $96 \times 96 \times 96$ using the combustion (MF) data set. The rendering results are shown in Figure 3. We can see that the rendered image has more artifacts with a subvolume size of $32 \times 32 \times 32$. For example, the yellow part at the top-left corner and the green part at the bottom-left corner are not smooth enough as shown in Figure 3 (a). The PSNR and SSIM curves are shown in Figure 2 (b). We can observe that using a larger subvolume size improves PSNR since an enlarged receptive field helps TSR-TVD capture more semantic information. However, a larger subvolume size takes more time to train and consumes more computing resources, as shown in Figure 2 (c). However, SSIM does not improve significantly as we update the subvolume size from $64 \times 64 \times 64$ to $96 \times 96 \times 96$. As a tradeoff, we use the subvolume size of $64 \times 64 \times 64$ to train TSR-TVD.

Number of ConvLSTM layers vs. PSNR and SSIM. We study the performance of TSR-TVD under different numbers of ConvLSTM layers using the combustion (HR) data set. We choose 1, 3, and 5 ConvLSTM layers to train TSR-TVD and plot the average PSNR and SSIM curves, as shown in Figure 2 (d). We can observe that using more ConvLSTM layers achieves higher PSNR, however, this parameter influences little the actual rendering results as the average SSIM value does not exhibit a significant difference. Moreover, using more ConvLSTM layers requires more computing resources. As a trade-off, we use one ConvLSTM layer to train TSR-TVD.

Maximal interpolation step vs. visual quality. At the inference stage, we use different numbers of maximal interpolation step K to interpolate the intermediate volumes using the supernova (VM) and vortex data sets. The rendering results are shown in Figure 4. For the supernova data set, we can observe that as K increases, the visual content gradually warps, although not by a large margin. As for the vortex data set, it is clear that different blue and red components grow or shrink as we increase K. The appropriate value for K is 5 or 7 for most of the data sets we explore. As these data sets were likely output sparsely from the simulations (refer to the accompanying video), the appropriate value for K would be proportionally larger than suggested here if we consider the actual simulation time steps.

Voxel shuffle / DeConv layer vs. training time. We investigate the training speed based on different upscaling methods for TSR-TVD using the vortex data set. The average training time curves are shown in Figure 2 (e). It is clear that using voxel shuffle layer can accelerate the training process. This is because voxel shuffle operation only increases the channel numbers in a tensor rather than the tensor size. Compared with changing the tensor size for upscaling, increasing the channel numbers uses GPU resources more efficiently and reduces the number of multiplications.

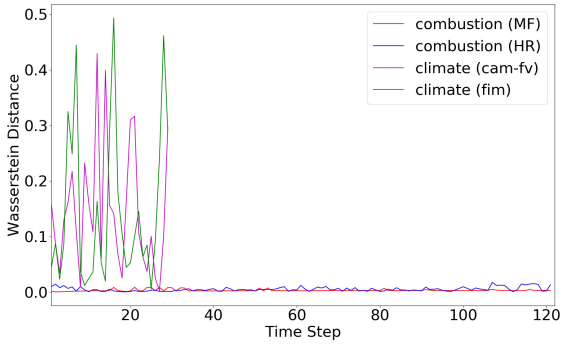


Fig. 5. Comparison of Wasserstein distances using the climate (cam-fv and fim) and combustion (HR and MF) data sets.

3 Additional Discussion

Failure case. The failure case for the climate (fim) data set as shown in Figure 10 in the paper is due to the limitation of TSR-TVD in estimating the difference between data distributions. For verification, we use Wasserstein distance [40] to measure the similarity of two neighboring time steps from two different data sets, as shown in Figure 5. We observe that the combustion (HR and MF) data set shows small distances between two neighboring time steps. For the climate (cam-fv and dim) data set, however, the distance fluctuates drastically. Moreover, the average Wasserstein distance for the climate (fim) data set is 0.14702, while that for the combustion (MF) data set is 0.00347. This estimation shows the limitation of TSR-TVD in capturing the fluctuation between data distributions.

Inference time. The inference time reported in Section 4.3 in the paper is for the entire remaining volume sequence (i.e., all time steps excluding those used for training). Take the combustion data set for example, during training, we crop a total of $19,200$ subvolumes. This is because we train 100 epochs; and in each epoch, we crop 4 subvolumes for each volume pair; and we have 48 volume pairs for training ($1 \sim 5, 2 \sim 6, \ldots, 48 \sim 52$). During inference, we need to crop $170,100$ subvolumes in order to generate the intermediate volumes. This is because we have 18 volume pairs for inference ($52 \sim 56, 56 \sim 60, \ldots, 118 \sim 122$); and for each pair, we crop 30 times along the x dimension, 45 times along the y dimension, and 7 times along the z dimension. Note that these cropped subvolumes have overlap in space in order to avoid spatial discontinuity. The inferred subvolumes are then concatenated through a weighted concatenation algorithm to form the whole volume.