V2V: A Deep Learning Approach to Variable-to-Variable Selection and Translation for Multivariate Time-Varying Data

Jun Han, Hao Zheng, Yunhao Xing, Danny Z. Chen, *Fellow, IEEE* and Chaoli Wang, *Senior Member, IEEE*

Abstract—We present V2V, a novel deep learning framework, as a general-purpose solution to the variable-to-variable (V2V) selection and translation problem for multivariate time-varying data (MTVD) analysis and visualization. V2V leverages a representation learning algorithm to identify transferable variables and utilizes Kullback-Leibler divergence to determine the source and target variables. It then uses a generative adversarial network (GAN) to learn the mapping from the source variable to the target variable via the adversarial, volumetric, and feature losses. V2V takes the pairs of time steps of the source and target variable as input for training, Once trained, it can infer unseen time steps of the target variable given the corresponding time steps of the source variable. Several multivariate time-varying data sets of different characteristics are used to demonstrate the effectiveness of V2V, both quantitatively and qualitatively. We compare V2V against histogram matching and two other deep learning solutions (Pix2Pix and CycleGAN).

Index Terms—Multivariate time-varying data, variable selection and translation, generative adversarial network, data extrapolation

1 Introduction

To understand various physical and natural phenomena, scientists produce *multivariate time-varying data* (MTVD) from large-scale scientific simulations. For large-scale simulations, the limited I/O bandwidths cannot match the rate of data production. In most cases, scientists could only afford to sparsely store the outputs for post hoc analysis and visualization. In this paper, we focus on the *variable-to-variable* (V2V) translation as an *extrapolation* task for MTVD. That is, given a variable sequence, for example, variable MF of the combustion data set, we aim to generate another variable sequence, for example, variable YOH of the same data set.

For scientific applications, it is meaningful to generate one variable sequence conditioned on another variable sequence because of the following reasons. First, scientists often simulate a large number of ensemble runs for generating multiple MTVD sequences but are only allowed to store a fraction of these sequences. Our solution allows saving one ensemble run entirely (i.e., all the time steps) while sparsely sampling the rest of runs (e.g., only the early time steps) as they can be faithfully recovered later on. In this way, scientists can save more runs, supporting a more accurate examination of the dynamic features of MTVD. Second, through V2V translation, scientists can discover the relationships among different variables and focus on variables of interest during post hoc investigation. As variable sequences depend on each other, studying the difficulty of transferring one sequence to another will shed new light on MTVD analysis and visualization.

Translating one variable sequence to another variable sequence poses four main challenges. First, understanding the relationships among different variables in MTVD is critical for V2V translation. Choosing two arbitrary variables for translation could lead to unexpected results since the randomly chosen variables may exhibit dramatically different patterns. Therefore, *variable selection* should be considered so that high-quality V2V translation can be achieved. Second, once the transferable variables are determined, choosing the appropriate source and target variables is still crucial since the translation difficulty could vary given a different variable as input. Third, unlike volume temporal and spatial super-resolution tasks that aim to interpolate data through their neighborhood information, V2V translation performs *extrapolation* instead of *interpolation*, which is a much more difficult task. Fourth, both *global* and *local* information must be considered simultaneously as multivariate temporal patterns in different regions are non-linear and non-uniform. Assuming the translation is local and linear may not ensure acceptable results: we may produce blurred features, resulting in fewer details in the visualization (i.e., direct volume rendering and isosurface rendering).

To tackle these challenges, we propose a novel solution for addressing the V2V translation problem for MTVD analysis and visualization, inspired by image-to-image translation tasks and representation learning techniques. V2V is a comprehensive framework for selecting transferable variables and synthesizing variable sequences. We leverage *generative adversarial networks* (GANs) to learn the variable mapping non-linearly and non-locally. Our solution consists of three stages: *feature learning* (aiming to find the relationships among different variables for MTVD), *translation graph construction* (aiming to detect the source and target variables), and *variable translation* (aiming to learn a mapping function from one variable to another variable). The training data could be obtained at earlier time steps from the two variable sequences. During inference, V2V can synthesize a variable sequence conditioned on another variable at later time steps. Quantitative and qualitative results with several data sets with different characteristics demonstrate the effectiveness of V2V. Also, we compare V2V against three other solutions: histogram matching [35], Pix2Pix [19], and CycleGAN [47]. Our results indicate that V2V achieves better quality using the *data*-, *image*-, and *feature*-level quality measures.

We summarize our contributions as follows. First, V2V is the first work in scientific visualization that applies deep learning techniques for variable selection and translation. Second, we propose a new architecture for the V2V translation task, which is different from the ones often used in the image-to-image translation task. Third, we conduct a thorough experiment to investigate how variable selection results could impact variable translation results.

2 Related work

Multivariate relationships. Researchers have studied point-wise correlation coefficients [8, 32, 38, 3] and gradient similarity measure [36]. Wang et al. [41] studied the information flow between variable pairs using transfer entropy for investigating variable causal relationships. Biswas et al. [2] classified variables using *surprise* and *predictability* derived from information theory and leveraged a graph-based representation for variable exploration. Liu et al. [26] designed the probabilistic association graph based on the *informativeness* and *uniqueness* concepts to uncover the hidden associations between different variables. Tao et al. [39] considered isosurface similarities across the time and variable dimensions for time-varying multivariate data and designed the matrix of isosurface similarity map for visual exploration.

- *J. Han, H. Zheng, D. Z. Chen, and C. Wang are with the University of Notre Dame. E-mail: {jhan5, hzheng3, dchen, chaoli.wang}@nd.edu.*
- *Y. Xing is with Sichuan University. E-mail: yhxing98@gmail.com.*

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/

Unlike these works, we investigate variable relationships using their latent features learned from a neural network and select variable pairs suitable for subsequent translation study.

Deep learning for scientific visualization. As deep learning solutions have solved a series of problems in image classification, image super-resolution, and image generation, researchers have recently attempted to explore the use of deep learning in solving scientific visualization problems. Examples include super-resolution generation in the spatial [46, 43, 9], temporal [13], and image [42] domains, volume rendering pipeline replacement [1, 17, 37], data reconstruction [11], workload balancing [18], and ensemble parameter space exploration [16]. Even though *interpolation* tasks have been investigated [46, 13, 9], in the context of MTVD *extrapolation*, no work has been done that synthesizes a variable sequence conditioned on another sequence, which is accomplished by this work.

Representation learning. Representation learning is a focused goal of many deep learning solutions. For example, Girdhar et al. [7] utilized an autoencoder to learn representative features of 3D objects for producing novel 3D objects and the corresponding 2D images. Chen et al. [4] designed LassoNet that attempts to learn a latent mapping from viewpoint and lasso to point cloud regions for lasso selection of 3D point clouds. Han et al. [10] proposed FlowNet, an autoencoder that learns the latent representations of streamlines and stream surfaces for dimensionality reduction and representative selection. Porter et al. [31] established a CNN to select representative time steps for time-varying multivariate data. Instead of selecting representative data or patches, V2V aims to find relationships among different variables in MTVD. Moreover, the learned relationships can guide us in choosing transferable variables for V2V translation.

Paired image-to-image translation. Deep learning solutions have achieved great success in image-to-image translation tasks, such as super-resolution and colorization. For image super-resolution, Dong et al. [6] proposed a CNN that learns the mapping from low-resolution images to high-resolution images. Johnson et al. [20] designed a CNN that simultaneously processes image style transfer and super-resolution tasks by minimizing content and style losses. Ledig et al. [22] presented a GAN for inferring photo-realistic high-resolution images from low-resolution images via optimizing adversarial and perceptual losses. For image colorization, Zhang et al. [44] designed a deep learning solution that produces vibrant and realistic colorful images conditioned on gray-scale images. Zhang et al. [45] established a CNN which directly maps a gray-scale image, along with sparse, local user "hints" to an output colorization and propagates user edits. Isola et al. [19] utilized conditional GANs for studying various image-to-image translation problems, such as aerial-to-map, day-to-night, and edge-to-photo. Park et al. [29] proposed a spatially-adaptive normalization layer for synthesizing photo-realistic images based on a semantic layout. Unlike image-to-image translation, in the context of scientific visualization, V2V establishes a variable selection process for choosing appropriate variable pairs for translation.

3 V2V

Let us denote $\mathbf{V}^{\text{var}} = \{\mathbf{V}^{\text{var}_1}, \mathbf{V}^{\text{var}_2}, \cdots, \mathbf{V}^{\text{var}_m}\}$ as a set of variables in the given MTVD, and $\mathbf{V}^{\text{var}_i} = \{\mathbf{V}_1^{\text{var}_i}, \mathbf{V}_2^{\text{var}_i}, \cdots, \mathbf{V}_n^{\text{var}_i}\}$ as the temporal sequence of variable i, where m is the number of variables and n is the number of time steps. $\mathbf{V}^{\text{var}_i}[1:k]$ is a subset of $\mathbf{V}^{\text{var}_i}$, which has the first k time steps ($n \gg k$). $\mathbf{V}^{\text{var}_i}[1:k]$ is also the samples we take to train our deep learning model. $\mathbf{V}^T$ and $\mathbf{V}^S$ denote, respectively, the ground truth (GT) and synthesized variables from V2V. $\mathbf{F}_j^{\text{var}_i}$ is the feature of variable i at the jth time step. Finally, let L, H, and W be the spatial dimensions of $\mathbf{V}^{\text{var}}$.

Our goal is to learn a mapping function $\mathscr{T}$ from one variable sequence $\mathbf{V}^{\text{var}_a}$ to another variable sequence $\mathbf{V}^{\text{var}_b}$, namely, $\mathbf{V}^{\text{var}_b} = \mathscr{T}(\mathbf{V}^{\text{var}_a})$. As sketched in Figure 1 (a), our approach consists of three stages: *feature learning*, *translation graph construction*, and *variable translation*. At the feature learning stage, we collect the available time steps from all variables of the given MTVD and utilize a U-Net [33] to learn their latent features. Then we leverage t-SNE [40] for dimensionality reduction where the latent feature of per variable and per time step

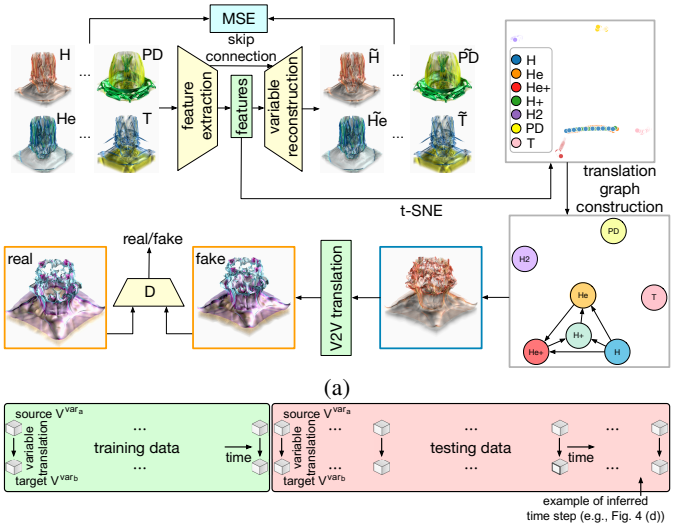


Fig. 1: (a) Overview of V2V. For feature learning, a U-Net is applied to extract features from variables and t-SNE is used to project the features for estimating variable similarity. A translation graph is constructed based on the learned variable features. For variable translation, variable pairs are selected and V2V is trained for learning the translation mapping. (b) Training and testing data from the volume sequence.

is projected onto a 2D space. The t-SNE projection helps us analyze and understand the similarities and differences among these variables, which provide us hints on whether or not a given pair of variables is transferable. Among all the variables, we select a transferable variable group (e.g., $\{\mathbf{V}^{\text{var}_1}, \cdots, \mathbf{V}^{\text{var}_p}\}$), where $p \leq m$. As an example, the transferable variable group for the example shown in Figure 1 is $\{\text{H}, \text{H+}, \text{He}, \text{He+}\}$.

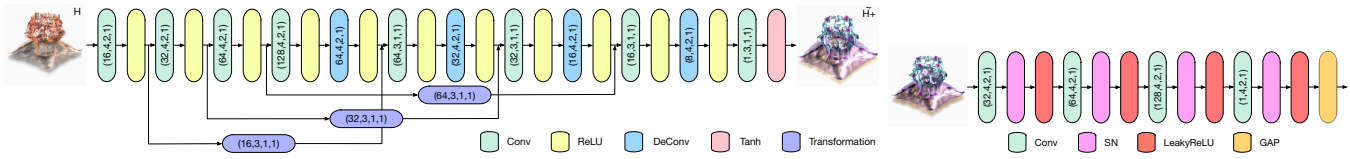
At the translation graph construction stage, given the transferable variable group, we estimate the transferable difficulty of variable b conditioned on variable a, and construct a translation graph $\mathscr{G}$ based on the computed transferable difficulty among different variables. Then, the source variable and target variable are selected from $\mathscr{G}$.

At the variable translation stage, we train a V2V network to learn the mapping between the two variable sequences (i.e., $\mathbf{V}^{\text{var}_a} \rightarrow \mathbf{V}^{\text{var}_b}$) based on the translation graph result. Our V2V includes one generator (G) and one discriminator (D). G consists of three modules: *feature extraction*, *feature translation*, and *variable translation*. The feature extraction module extracts rich semantic information from the input variable. The feature translation module translates the features from the source variable to the target variable at different scales. The variable translation module translates the refined features to the target variable domain. As shown in Figure 1 (b), in our experiments, the training data consist of early time steps of $\mathbf{V}^{\text{var}_a}$ and $\mathbf{V}^{\text{var}_b}$, and the testing data consist of later time steps of $\mathbf{V}^{\text{var}_a}$ and $\mathbf{V}^{\text{var}_b}$.

For U-Net training, we utilize the mean squared error (MSE) as the loss function to compute the difference between the reconstructed and GT variables. For V2V training, we leverage *adversarial*, *volumetric*, and *feature* losses to optimize the network. Next, we describe our approach in detail, including the network architectures of feature learning (Section 3.1) and variable translation (Section 3.3), as well as the algorithm for constructing the translation graph (Section 3.2).

3.1 Feature Learning

At the feature learning stage, we leverage a U-Net that takes the available time steps of all variables from the same MTVD as input and outputs feature descriptors per variable and per time step. U-Net also allows the reconstruction of a variable at a time step from the corresponding feature descriptor. Skip connections in U-Net can bridge different semantic features of the same scale, avoiding information loss during reconstruction.



(a) Generator G (b) Discriminator D

Fig. 2: Network architecture of V2V. (a) G contains eight Conv layers, four DeConv layers, and three transformation blocks. (b) D includes four Conv layers, four SN layers, and one GAP layer.

Algorithm 1 Translation graph construction.

Require: A set of variables: $\{\mathbf{V}^{\text{var}_1}, \mathbf{V}^{\text{var}_2}, \cdots, \mathbf{V}^{\text{var}_p}\}$.
initialize a translation graph $\mathscr{G}$ with $\text{var}_1, \cdots, \text{var}_p$ as nodes and no edge
for $i = 1 \cdots p$ **do**
  for $j = 1 \cdots i$ **do**
    if $\mathscr{E}(\mathbf{F}^{\text{var}_i}, \mathbf{F}^{\text{var}_j}) < \varepsilon$ **then**
      Compute $\text{TD}(\mathbf{V}^{\text{var}_j} || \mathbf{V}^{\text{var}_i})$ and $\text{TD}(\mathbf{V}^{\text{var}_i} || \mathbf{V}^{\text{var}_j})$
      if $\text{TD}(\mathbf{V}^{\text{var}_j} || \mathbf{V}^{\text{var}_i}) < \text{TD}(\mathbf{V}^{\text{var}_i} || \mathbf{V}^{\text{var}_j})$ **then**
        add an edge from var_i to var_j to $\mathscr{G}$
      else
        add an edge from var_j to var_i to $\mathscr{G}$
      end if
    end if
  end for
end for
return $\mathscr{G}$

In general, U-Net is composed of an *encoding path* and a *decoding path*. There are four convolutional (Conv) layers in the encoding path and four composites of deconvolutional (DeConv) layers and Conv layers in the decoding path. In the encoding path, each of the first three Conv layers reduces the input's dimension by half. The feature maps start with 64 and double in the following Conv layers. Then, we apply one Conv layer to transform the learned features into a 1D vector with 512 components. In the decoding path, DeConv layers are utilized to upscale the feature back to the original dimension, and the following Conv layers are utilized to fuse and refine feature maps. To keep the information flowing smoothly and avoid information loss, we concatenate the feature maps from the Conv layer in the encoding path and the feature maps from the DeConv layer as input for the consecutive Conv layer for refinement. The feature maps start with 256 and reduce by half in the following DeConv layers. We keep the same feature maps in each Conv layer followed by each DeConv layer. Note that the concatenation happens at the corresponding scale (i.e., these two tensors have the same resolution). Rectified linear unit (ReLU) [28] is utilized after each Conv or DeConv layer to help the network learn faster and perform better. After the final Conv layer, $\tanh(\cdot)$ is applied for normalization (in the range of $[-1, 1]$).

Loss function. In order to ensure that the synthesized variables are close to the GT variables, we use MSE as the loss function to train U-Net. The MSE loss is defined as

$$\mathscr{L} = \sum_{j=1}^{u} ||\hat{\mathbf{V}}_j - \mathbf{V}_j||_2, \tag{1}$$

where $\mathbf{V}_j$ and $\hat{\mathbf{V}}_j$ are, respectively, the GT and synthesized variables of the jth training sample, $u = k \times m$ is the number of training samples, and $||\cdot||_2$ is L_2 norm.

3.2 Translation Graph Construction

After the transferable variable group (e.g., $\{\mathbf{V}^{\text{var}_1}, \cdots, \mathbf{V}^{\text{var}_p}\}$) is determined, we define the *transferable difficulty* (TD) of $\mathbf{V}^{\text{var}_j}$ conditioned on $\mathbf{V}^{\text{var}_i}$ as follows

$$\text{TD}(\mathbf{V}^{\text{var}_j} || \mathbf{V}^{\text{var}_i}) = \frac{1}{k} \sum_{t=1}^{k} \text{KL}(\mathbf{F}_t^{\text{var}_j} || \mathbf{F}_t^{\text{var}_i}), \tag{2}$$

where $\mathbf{F}_t^{\text{var}_j}$ is the feature of variable j at time step t, $\text{KL}(\cdot||\cdot)$ is Kullback-Leibler divergence, and k is the total available time steps. The transferable order for the variable pair i and j is given by

$$\min\{\text{TD}(\mathbf{V}^{\text{var}_j} || \mathbf{V}^{\text{var}_i}), \text{TD}(\mathbf{V}^{\text{var}_i} || \mathbf{V}^{\text{var}_j})\}. \tag{3}$$

A translation graph $\mathscr{G}$ can be constructed based on the calculation of TDs between different variable pairs. The process is described in Algorithm 1. Given a pair of variables, we first compute the Euclidean distance of these two variables in the feature space. The distance determines whether or not these two variables are transferable. If the distance is less than a threshold ε, then we compute TD for the two variables, and determine the transferable order based on Equation 3.

3.3 Variable Translation

Generator. G consists of three modules. The feature extraction module utilizes four Conv layers to extract the features from the input variable. Each Conv downscales the input by half, and a ReLU is followed to accelerate the training and improve model performance. The feature translation module leverages three *transformation blocks* to translate the features from the source variable to the target variable at different scales and feed into the variable translation module. Each transformation block includes two paths. One path contains three Conv layers, and the other path contains one Conv layer. Finally, these two paths are connected by addition [15]. The variable translation module utilizes four DeConv layers and four Conv layers followed by ReLU to map the feature to the output variable domain. Each DeConv upscales the input twice. In addition, after each DeConv layer, we stack the outputs from DeConv and the feature translation stage together and feed into a Conv layer. The two stacked outputs share the same scale, and the Conv layer does not change the scale of the input. Note that we only use $\tanh(\cdot)$ in the final Conv layer. The architecture of G is sketched in Figure 2 (a).

Discriminator. D includes four Conv layers, four spectral normalization (SN) [27] layers, and one global average pooling (GAP) [25] layer. Each Conv downscales the input by half, and SN is followed by Conv to normalize the weights in Conv for training stabilization. Leaky ReLU activation ($\alpha = 0.2$) is applied after each Conv layer. Finally, one GAP is leveraged to squeeze the output into a tensor with $1 \times 1 \times 1 \times 1$. No activation function is added after GAP. The architecture of D is sketched in Figure 2 (b).

Loss function. As suggested by Han and Wang [13], we apply adversarial, volumetric, and feature losses to optimize V2V so that the synthesized variables are close to the GT variables. The adversarial loss is defined as

$$\min_{\theta_G} \mathscr{L}_G = \mathbb{E}_{V \in \mathbf{V}^S}\left[(D(G(V)) - 1)^2\right], \tag{4}$$

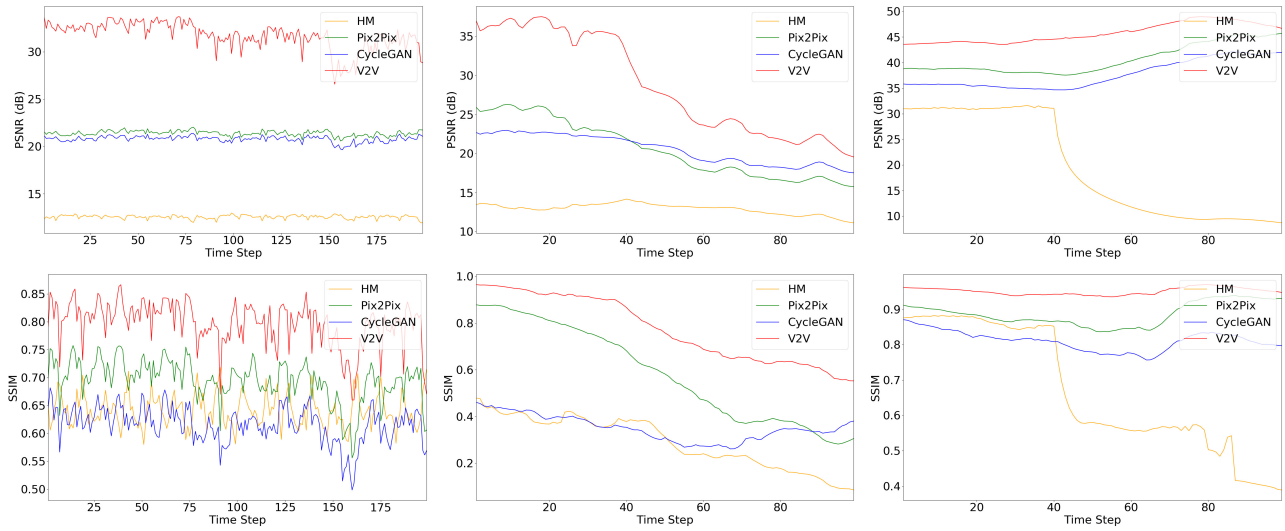
$$\min_{\theta_D} \mathscr{L}_D = \frac{1}{2}\mathbb{E}_{V \in \mathbf{V}^S}\left[D(V)\right] + \frac{1}{2}\mathbb{E}_{V \in \mathbf{V}^T}\left[(D(G(V)) - 1)^2\right], \tag{5}$$

where θ_G and θ_D are the learnable parameters in G and D, and $\mathbb{E}[\cdot]$ denotes the expectation operation.

The volumetric loss is defined as

$$\mathscr{L}_V = \mathbb{E}_{V' \in \mathbf{V}^S, V \in \mathbf{V}^T}\left[||G(V') - V||_2\right], \tag{6}$$

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



(a) climate (SALT→TEMP) (b) combustion (MF→YOH) (c) ionization (H→H+)

Fig. 3: PSNR (top row) and SSIM (bottom row) of synthesized variables (TEMP, YOH, and H+) under HM, Pix2Pix, CycleGAN, and V2V.

The feature loss is defined as

$$\mathscr{L}_F = \mathbb{E}_{V' \in \mathbf{V}^s, V \in \mathbf{V}^T} \sum_{k=1}^{N} \frac{1}{N_k} \left[||F^k(G(V')) - F^k(V)||_1 \right], \quad (7)$$

where N is the total number of Conv layers in D, N_k is the number of elements in the kth Conv layer, and $F^k(\cdot)$ denotes the feature representation at the kth Conv layer.

The overall loss for G is the combination of the three losses

$$\min_{\theta_G} \mathscr{L}_G = \lambda_1 \left(\mathbb{E}_{V \in \mathbf{V}^T} \left[\left(D(G(V)) - 1 \right)^2 \right] \right) + \lambda_2 \mathscr{L}_V + \lambda_3 \mathscr{L}_F, \quad (8)$$

where λ_1, λ_2, and λ_3 are weights, each in the range of $[0, 1]$.

Note that adversarial, volumetric, and feature losses serve different purposes in V2V training. Adversarial loss aims to judge the realness of the synthesized volumes from the generator. Volumetric loss seeks to ensure that the synthesized volumes are close to the GT volumes. Feature loss aims to stabilize the training process and enhance the visual quality.

Table 1: The variables and dimensions of each data set.

data set	variables	dimension ($x \times y \times z \times t$)
climate	SALT, TEMP	$360 \times 66 \times 27 \times 200$
combustion	CHI, HR, MF, YOH	$480 \times 720 \times 120 \times 100$
ionization	H, H+, He, He+, H2, PD, T	$600 \times 248 \times 248 \times 100$

Table 2: Average PSNR and SSIM values. The best ones are highlighted in bold.

data set ($v_1 \rightarrow v_2$)	method	PSNR (dB)	SSIM
climate (SALT→TEMP)	HM	13.12	0.642
	Pix2Pix	21.39	0.695
	CycleGAN	20.78	0.616
	V2V	**31.69**	**0.797**
combustion (MF→YOH)	HM	12.96	0.291
	Pix2Pix	20.46	0.585
	CycleGAN	20.56	0.351
	V2V	**28.73**	**0.776**
ionization (H→H+)	HM	19.62	0.668
	Pix2Pix	40.58	0.887
	CycleGAN	37.59	0.812
	V2V	**45.75**	**0.951**

4 Results and Discussion

4.1 Data Sets and Network Training

We experimented with our approach using the data sets listed in Table 1. We implemented V2V based on PyTorch [30] and used a single NVIDIA TESLA P100 GPU for training. For feature learning, we used the bicubic kernel with a downscaling factor of four to downscale combustion and ionization data sets for fast training. For variable translation, we used the original resolution for training; however, for each epoch, we randomly crop the volumes. This cropping mechanism can reduce training cost and GPU memory consumption. We point out that V2V can be applied to volumes of arbitrary size because it is fully convolutional. We scaled the range of $\mathbf{V}^{\text{var}}$ to $[-1, 1]$. All learnable parameters in U-Net and V2V are initialized using He et al. [14] and the Adam algorithm [21] is applied for parameter update. We set one training sample per mini-batch. For training U-Net, the learning rate is set to 10^{-4}. For training V2V, different learning rates for G and D are set as suggested by Roth et al. [34]. 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 $\lambda_3 = 5 \times 10^{-1}$. We trained U-Net and V2V for 50 and 150 epochs for all data sets, respectively. We sampled the first 40% time steps for training and the rest for inference. All these hyperparameters are determined based on experiments.

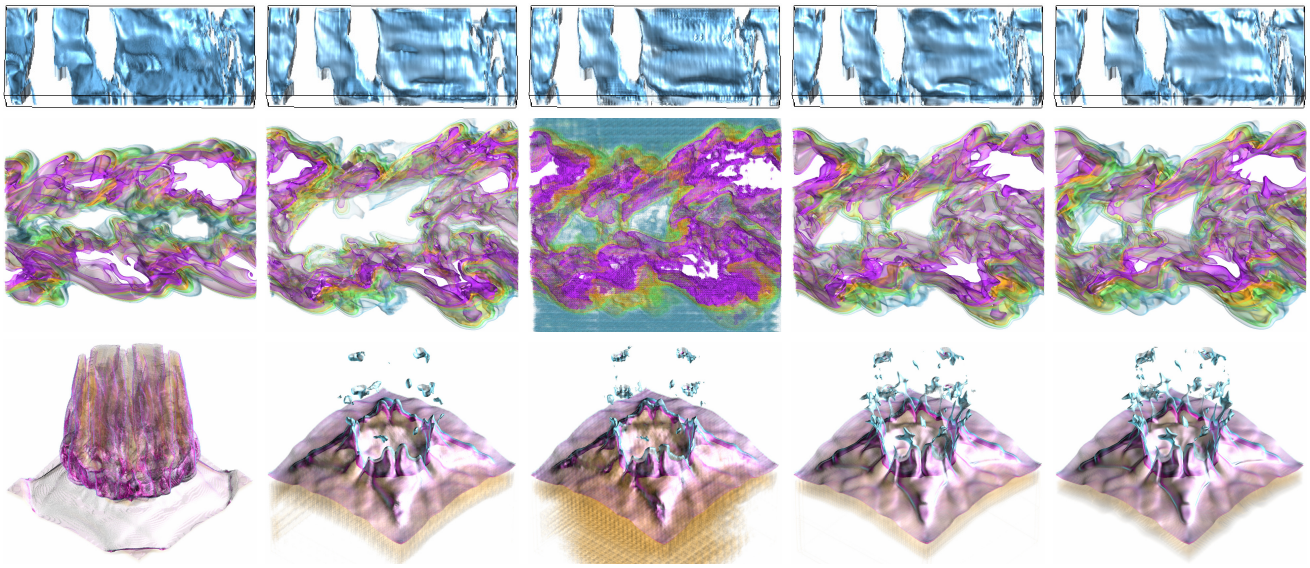
4.2 Results

Baselines. We compare one baseline solution for variable selection:

- Biswas et al. [2]: It is an information-theoretic approach for variable grouping. Once grouped, users can select representative variables for further exploration. We leverage this solution to select variable pairs as the input to the V2V translation task.

Note that Biswas et al. is a solution for variable selection *only*, and *not* for variable translation. For translation comparison, we implement three baseline solutions for the V2V translation task:

- Histogram matching (HM) [35]: HM is a traditional approach for translating one data set to another one conditioned on the content and style of the data. We apply HM to translate variable j at time step k conditioned on variable i at time step k and variable j at time step $k-1$.
- Pix2Pix [19]: Pix2Pix is the first *paired* image-to-image translation framework. The original Pix2Pix architecture is leveraged for the V2V translation task.
- CycleGAN [47]: CycleGAN is a deep learning solution for *unpaired* image-to-image translation. Since the variables are paired in our V2V translation task, we replace the *identity* loss in CycleGAN with the *volumetric* loss in V2V.



(a) HM (b) Pix2Pix (c) CycleGAN (d) V2V (e) GT

Fig. 4: Comparison of volume rendering results. Top to bottom: the climate (SALT→TEMP), combustion (MF→YOH), and ionization (H→H+) data sets. Displayed here are the renderings of TEMP at time step 159, YOH at time step 65, and H+ at time step 70, respectively.

Table 3: Average IS values at chosen isovalues. The best ones are highlighted in bold.

	HM		Pix2Pix		CycleGAN		V2V	
data set ($v_1 \rightarrow v_2$)	$v=-0.4$	$v=0.3$	$v=-0.4$	$v=0.3$	$v=-0.4$	$v=0.3$	$v=-0.4$	$v=0.3$
climate (SALT→TEMP)	0.12	0.15	0.83	0.85	0.73	0.79	**0.92**	**0.91**
	$v=-0.9$	$v=0.65$	$v=-0.55$	$v=-0.9$	$v=-0.55$	$v=-0.9$	$v=-0.9$	$v=-0.55$
combustion (MF→YOH)	0.23	0.19	0.72	0.69	0.47	0.49	**0.82**	**0.84**
	$v=-0.96$	$v=-0.9$	$v=-0.96$	$v=-0.9$	$v=-0.96$	$v=-0.9$	$v=-0.96$	$v=0.9$
ionization (H→H+)	0.32	0.41	0.82	0.84	0.79	0.81	**0.92**	**0.95**

Table 4: Total training time (in hour), average inference time (in second), and model size (MB) under Pix2Pix, CycleGAN, and V2V.

data set	method	training epochs	training time	inference time	model size
climate	Pix2Pix	150	6.79	3.52	6
	CycleGAN	200	56.44	4.63	26
	V2V	150	15.36	4.01	14
combustion	Pix2Pix	150	31.21	187.45	6
	CycleGAN	200	169.08	220.36	26
	V2V	150	53.14	194.72	14
ionization	Pix2Pix	150	21.43	122.39	6
	CycleGAN	200	125.46	130.93	26
	V2V	150	40.76	129.07	14

Table 5: Average PSNR and SSIM values for variable translations using Biswas et al. [2] and V2V. The better ones are highlighted in bold.

data set	variable pair	approach	PSNR (dB)	SSIM
combustion	YOH→CHI	Biswas et al.	24.76	0.607
	MF→CHI	V2V	**35.76**	**0.829**
ionization	T→H+	Biswas et al.	33.41	0.827
	H→H+	V2V	**45.75**	**0.951**

For a fair comparison, we use the same loss functions (i.e., adversarial, volumetric, and feature losses) designed for V2V to train Pix2Pix and CycleGAN.

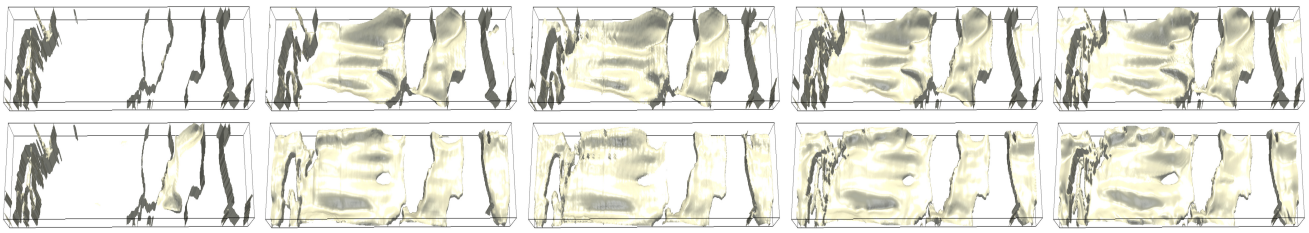
Due to the page limit, we show the frame-to-frame comparison results in the accompanying video. Unless otherwise stated, we display all visualization results using the inferred volumes (refer to Figure 1 (b) for an example). The same settings for lighting, viewing, transfer function (for direct volume rendering), and isovalue (for isosurface rendering) are applied to all visualization results for the same data set. With respect to the GT, we compare our V2V results with those of HM, Pix2Pix, and CycleGAN.

Table 6: Average PSNR and SSIM values for different V2V translations of the ionization data set. H is the source variable.

target variable	PSNR (dB)	SSIM
H+	45.75	0.951
He	37.44	0.837
He+	39.99	0.874
PD	31.68	0.616

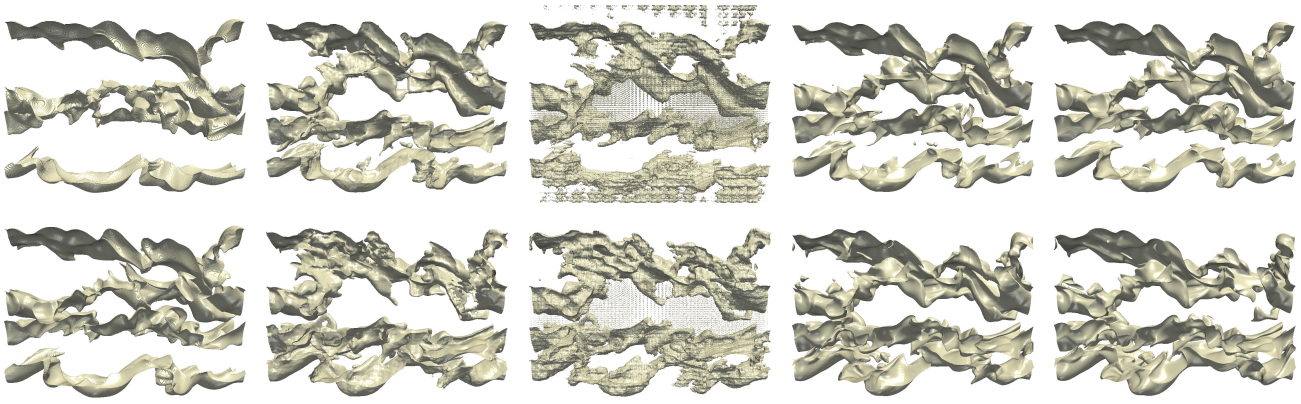
Evaluation metrics. For quantitative evaluation, we compute, between the synthesized variables and GT variables, the *peak signal-to-noise* (PSNR) at the data level, *structural similarity index* (SSIM) at the image level, and *isosurface similarity* (IS) [13] at the feature level.

Quantitative and qualitative analysis. In Figure 3, we show the data (PSNR) and image (SSIM) level results using HM, Pix2Pix, CycleGAN, and V2V. At the data level, for the climate (SALT→TEMP) data set, all four curves exhibit a periodic pattern since each time step denotes the temperature for each month and 12 time steps are for one year. The PSNR values of V2V outperform those of HM, Pix2Pix, and CycleGAN. For the combustion (MF→YOH) data set, PSNR values decrease as time step goes. This is because, at the later time steps, the temporal behavior becomes more turbulent and complex, making the prediction more difficult. Again, V2V still outperforms HM, Pix2Pix, and CycleGAN. For the ionization (H→H+) data set, it is clear that V2V achieves the highest PSNR values for each time step. At the image level, V2V can still produce higher SSIM values compared with HM, Pix2Pix, and CycleGAN. It is the clear winner for the climate (SALT→TEMP), combustion (MF→YOH), and ionization (H→H+) data sets. For the combustion (MF→YOH) data set, due to the increase of visual content, the SSIM values decrease as time step goes. In Table 2, the average PSNR and SSIM values for HM, Pix2Pix, CycleGAN, and V2V are reported. Again, V2V achieves the best PSNR



(a) HM (b) Pix2Pix (c) CycleGAN (d) V2V (e) GT

Fig. 5: Comparison of isosurface rendering results of the climate (SALT→TEMP) data set at time step 167. The chosen isovalues are $v = -0.4$ (top row) and $v = 0.3$ (bottom row).



(a) HM (b) Pix2Pix (c) CycleGAN (d) V2V (e) GT

Fig. 6: Comparison of isosurface rendering results of the combustion (MF→YOH) data set at time step 53. The chosen isovalues are $v = -0.9$ (top row) and $v = -0.55$ (bottom row).

and SSIM values. Note that the PSNR and SSIM curves of HM suddenly decrease after time step 40 for the combustion and ionization data sets since we only use 40% data for training. The error accumulates when predicting the later time steps. Since the climate data set is periodic, the PSNR and SSIM curves of HM do not exhibit a similar pattern as that of the other two data sets.

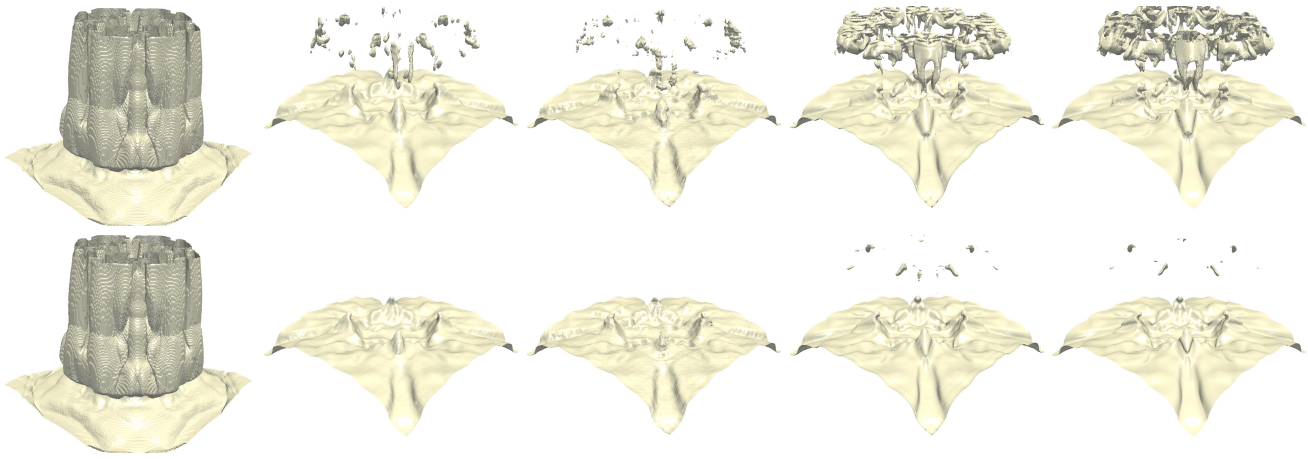
In Figure 4, the volume rendering results of the volumes synthesized by HM, Pix2Pix, CycleGAN, and V2V are shown. For the climate (SALT→TEMP) data set, the rendering results synthesized by Pix2Pix and CycleGAN contain artifacts. The result generated by HM cannot well capture the main structure, while the result produced by V2V is much smoother and similar to the GT. For the combustion (MF→HR) data set, V2V produces finer details with respect to GT, while HM and CycleGAN fail to recover the volume well. Pix2Pix generates some artifacts and is unable to recover the content around the volume boundary. For the ionization (H→H+) data set, V2V achieves the best result compared with HM, Pix2Pix, and CycleGAN. For example, for the Pix2Pix result, there are fewer details at the top part, and there are some artifacts at the bottom layer. For the CycleGAN result, it produces more orange content at the bottom part and fails to accurately recover the top part. For the HM result, it generates more purple and yellow content at the top part.

In Figures 5, 6, and 7, the isosurface rendering results of the volumes synthesized by HM, Pix2Pix, CycleGAN, and V2V using the climate (SALT→TEMP), combustion (MF→YOH), and ionization (H→H+) data sets are displayed. For each data set, we choose one time step and two isovalues to render the isosurfaces. For the climate (SALT→TEMP) data set, it is evident that V2V can generate the highest quality isosurfaces compared with HM, Pix2Pix, and CycleGAN. HM fails to construct the isosurfaces close to GT, and the isosurfaces extracted from Pix2Pix and CycleGAN are filled with noises and artifacts. Similar observations can be made for the combustion (MF→YOH) data set where V2V generates the highest quality isosurfaces compared with HM, Pix2Pix, and CycleGAN. For the ionization (H→H+) data set, V2V produces the highest quality isosurfaces. Pix2Pix and CycleGAN fail to construct the isosurface at the top part, and HM synthesizes fake features compared with the GT results. Furthermore, the average IS values for these three data sets are reported in Table 3. The average IS values also demonstrate that V2V achieves the best quality. Moreover, among Pix2Pix, CycleGAN, and V2V, CycleGAN has almost the worst performance in terms of PSNR, SSIM, and IS. This is because, unlike image-to-image translation, where the translation is symmetric (e.g., day to night), in V2V translation, the translation is asymmetric (e.g., it is more challenging to translate from CHI to MF compared with translating from MF to CHI). Therefore, adding cycle consistency will hurt the translation performance. As for Pix2Pix, this architecture is too simple to capture the complex structure changes between the variables.

In Table 4, we report the total training time (in hour), the average inference time (in second), and model size for Pix2Pix, CycleGAN, and V2V, respectively. As we can see, CycleGAN takes the longest training time since it needs to go through the network six times in one iteration (i.e., two discriminators, one cycle of X→Y→X, and another cycle of Y→X→Y). Pix2Pix and V2V only need to go through the network twice in one iteration (i.e., one discriminator and one generator). As for the inference time, there is no significant difference. In terms of model size, V2V needs 14MB to store parameters.

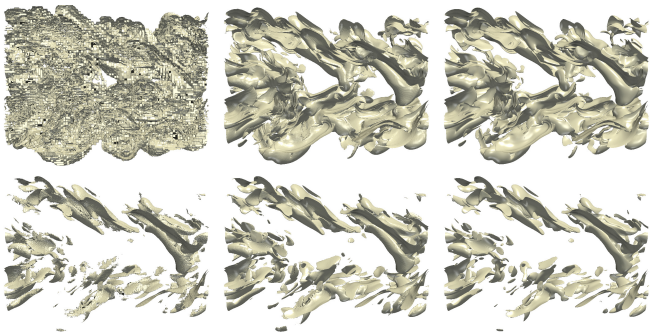
Comparison against compression. In Figure 8, we compare V2V and an advanced lossy compression (LC) method [24, 23] using isosurface rendering results. This LC method can achieve a high compression rate while producing less data distortion. To achieve a fair comparison, we set a similar PSNR value (i.e., 29.5 dB) for both approaches. As we can see, the isosurfaces generated by LC include significant noises and artifacts compared with those produced by V2V.

Evaluation of variable selection. To show the effectiveness of the proposed variable selection solution, we compare V2V against Biswas et al. [2]. We only use Biswas et al. to choose transferable variable pairs, as it does not perform variable translation. Once the pairs are selected from either V2V or Biswas et al. we apply the same solution (i.e., V2V) for translation. The training time of the variable selec-



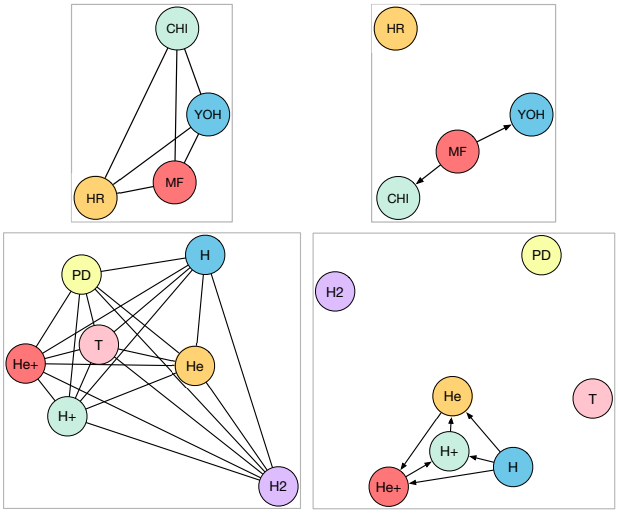
(a) HM (b) Pix2Pix (c) CycleGAN (d) V2V (e) GT

Fig. 7: Comparison of isosurface rendering results of the ionization (H→H+) data set at time step 92. The chosen isovalues are $v = -0.96$ (top row) and $v = -0.9$ (bottom row).



(a) LC [24, 23] (b) V2V (c) GT

Fig. 8: Isosurface rendering results using the combustion (CHI) data set at time step 60. The chosen isovalues are $v = -0.7$ (top row) and $v = 0.3$ (bottom row).



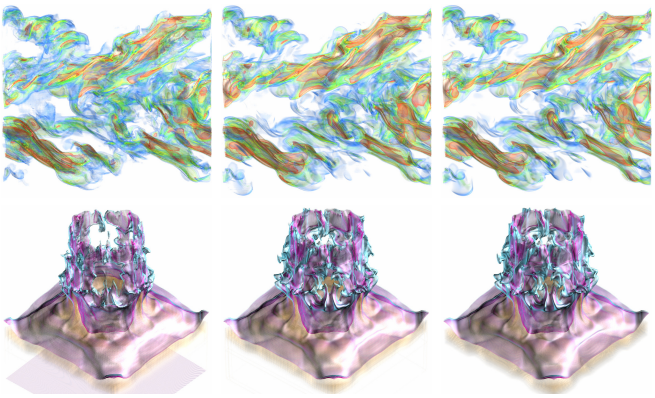
(a) Biswas et al. [2] (b) V2V

Fig. 9: Comparison of clustered graphs (left column) and translation graphs (right column). Top row: combustion. Bottom row: ionization. For both graphs, the distance between two variables in the 2D graph indicates their similarity.

tion stage for the combustion and ionization data sets is 1.86 and 2.14 hours, respectively. The training time depends on the number of variables and the dimension of the data set. In Figure 9, we show clustered graphs and translation graphs of the combustion and ionization data sets using Biswas et al. and V2V, respectively. Note that the clustered graphs of Biswas et al. are fully-connected and undirected, while the translation graphs of V2V are partially-connected and directed. For the combustion data set, Biswas et al. demonstrates that YOH and CHI are more similar compared with MF and CHI, while V2V leads to the opposite conclusion. For the ionization data set, Biswas et al. demonstrates that T and H+ are similar, while H and H+ are distinguishable; however, V2V gets the opposite results.

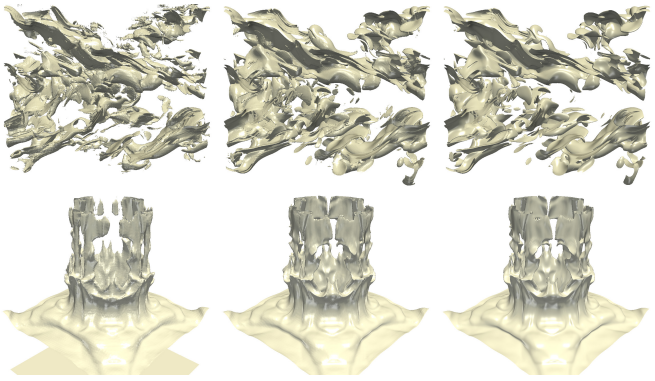
To evaluate the effectiveness of these two variable selection approaches, we choose two pairs from Biswas et al. (i.e., YOH→CHI and T→H+) and two from V2V (i.e., MF→CHI and H→H+) for the translation task. The results are shown in Figures 10 and 11. As we can see, for Biswas et al. YOH→CHI and T→H+ are not successfully judged from both volume and isosurface rendering results. For example, the volume rendering of CHI and H+ cannot exhibit a good visual quality compared with GT. As for the variable pairs selected by V2V, the translation results are satisfactory. Table 5 reports the average PSNR and SSIM values under these two variable translation schemes. Overall, based on the chosen source and target variables, V2V achieves higher PSNR and SSIM values in the translation task. These results indicate that, unlike V2V, variable pairs selected according to Biswas et al. may not be suitable for variable translation.

To further evaluate the effectiveness of the variable selection process, we use the ionization data set, choose H as the source variable, and translate it to H+, He, He+, and PD. In Figure 12, we show the volume rendering results. For H→H+, H→He, H→He+, the synthesized results are similar to GT. However, for H→PD, V2V fails to recover the details of PD, particularly, the structure of the top part is not captured. This failure may be explained by a large distance between H and PD shown in the translation graph (Figure 9 (b)). The isosurface rendering results are shown in Figure 13. For H→H+ and H→He, the isosurfaces generated by V2V are close to GT and almost exhibit the same volumetric features. For H→He+, V2V can still recover the isosurfaces but miss some details. For example, the detailed surface features at the top part are missing in the isosurface synthesized by V2V. For H→PD, V2V fails to generate high-quality isosurface compared with the GT isosurface. For example, the isosurface generated by V2V consists of noises and artifacts, and details are missing at the bottom layer. We also report the average PSNR and SSIM values in Table 6. The quantitative results also confirm the difficulty of translating H to PD. Based on the proposed solution, for the combustion set, scientists can save 60 time steps for the variables YOH and CHI if MF is the source variable, and 18.53GB storage is saved in total. As for the ionization, 21.05GB can be saved if H is the source variable since



(a) Biswas et al. [2] (b) V2V (c) GT

Fig. 10: Comparison of variable selection approaches via volume rendering. Variable pairs selected by Biswas et al. are YOH→CHI (top row) and T→H+ (bottom row). Variable pairs selected by V2V are MF→CHI (top row) and H→H+ (bottom row). The displayed time steps are 80 and 50 for CHI and H+, respectively.



(a) Biswas et al. [2] (b) V2V (c) GT

Fig. 11: Comparison of variable selection approaches via isosurface rendering. Variable pairs selected by Biswas et al. are YOH→CHI (top row) and T→H+ (bottom row). Variable pairs selected by V2V are MF→CHI (top row) and H→H+ (bottom row). The chosen isovalues are $v = -0.6$ (top row) for CHI and $v = -0.1$ (bottom row) for H+. The displayed time steps are 80 and 50 and for CHI and H+, respectively.

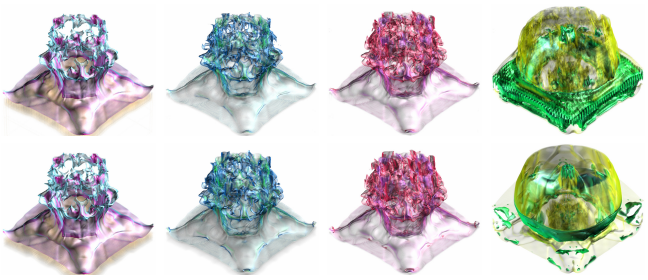
these variables (H+, He, and He+) are only stored 40 time steps.

Evaluation of variable order. To verify that the translation order does impact the translation performance, we use the combustion data set and choose two translations, MF→CHI and CHI→MF. The results are demonstrated in Figure 14. As we can see, CHI→MF is unsatisfactory since the synthesized isosurfaces fail to capture the interesting features compared with GT. However, MF→CHI is successful since the generated isosurfaces are very close to GT. This asymmetric translation is likely because the essential information in MF is richer than that in CHI, which makes MF→CHI easier than CHI→MF.

4.3 Hyperparameter Study

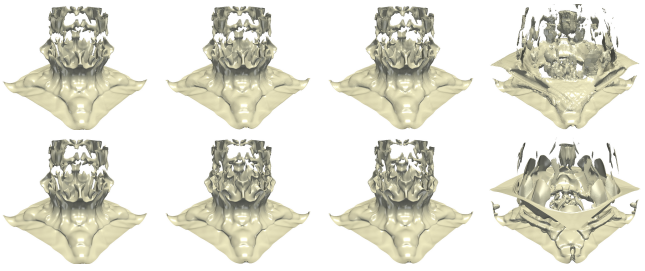
To evaluate V2V, we study these hyperparameter settings: training epochs, training samples, crop size, and feature translation module. The detailed discussion is given below.

Training epochs. We investigate how the increasing number of training epochs influences the rendering quality of the synthesized volumes. Isosurface rendering results obtained at different numbers of training epochs are shown in Figure 15 for the combustion (MF→YOH) data set. We can see that there are some artifacts at the bottom-right and top-left corners with 100 epochs, while these artifacts are eliminated with 150 epochs. Moreover, we observe that the PSNR values improve with 150 epochs. However, there is no significant difference between synthesized results between 100 and 150 epochs. So, we recommend using 150 epochs to train V2V.



(a) H+ (b) He (c) He+ (d) PD

Fig. 12: Comparison of volume rendering results of the ionization data set at time step 60. H is chosen as the source variable. Top row: V2V. Bottom row: GT.

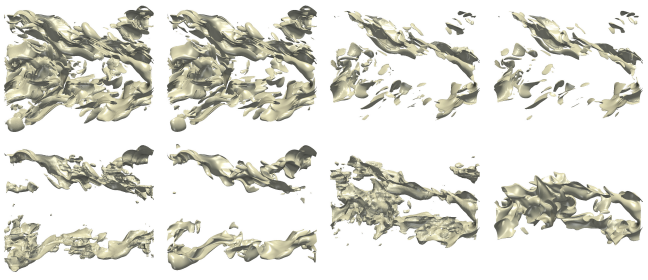


(a) H+ (b) He (c) He+ (d) PD

Fig. 13: Comparison of isosurface rendering results of the ionization data set at time step 60. H is chosen as the source variable. Top row: V2V. Bottom row: GT.

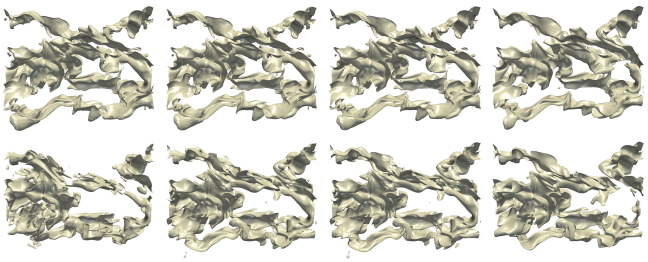
Training samples. We study how the number of training samples impacts visual quality, PSNR, and SSIM. 20%, 40%, and 60% training samples are applied to train V2V using the climate (SALT→TEMP) data set. As shown in Figure 16, only using 20% samples to train V2V could lead to some artifacts in volume rendering results while using 40% can mostly remove these artifacts. As for isosurface rendering, the isosurface result generated by using 20% samples could miss some details with $v = 0.45$. In addition, the average PSNR and SSIM curves under different training samples are displayed in Figure 17 (a). As we can observe, PSNR and SSIM values improve with the use of more training samples. But this comes with longer training time, as indicated in Figure 17 (b). We observe that beyond 40% samples, there is almost no improvement in visual quality. Hence, we suggest using 40% samples to train V2V.

Crop size. Due to the GPU memory constraint, V2V cannot process the whole scalar data at the same time. Therefore, we crop subvolumes to train V2V. We train V2V with subvolume sizes of 128×64^2,



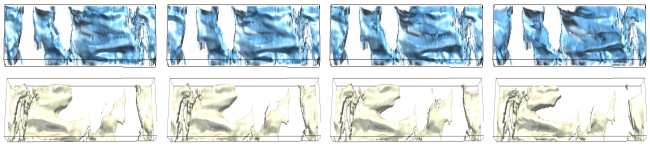
(a) V2V (b) GT (c) V2V (d) GT

Fig. 14: Evaluation of translation order using the combustion data set via isosurface rendering at time step 72. Top row: MF→CHI (TD = 7.10). Bottom row: CHI→MF (TD = 8.07). The chosen isovalues are $v = -0.6$ (1st and 2nd columns) and $v = 0.5$ (3rd and 4th columns).



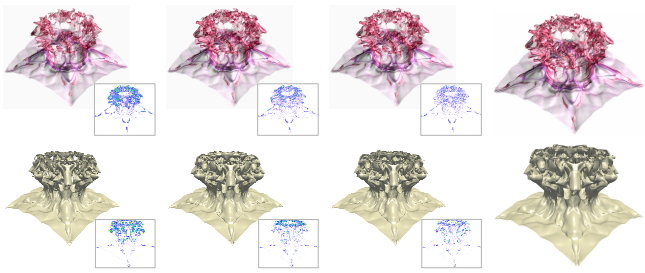
(a) 50 (b) 100 (c) 150 (d) GT

Fig. 15: Comparison of isosurface rendering results under different training epochs using the combustion (MF→YOH) data set at time step 70. The chosen isovalues are $v = -0.7$ (top row) and $v = -0.3$ (bottom row).



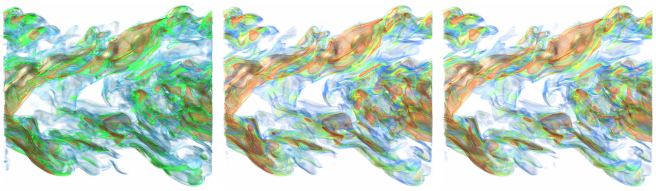
(a) 20% (b) 40% (c) 60% (d) GT

Fig. 16: Comparison of volume rendering (1st row) and isosurface (2nd row) rendering results under different training samples using the climate (SALT→TEMP) data set at time step 176. The chosen isovalue is $v = 0.45$ (2nd row).



(a) 128×64^2 (b) 192×96^2 (c) 256×128^2 (d) GT

Fig. 18: Comparison of volume rendering (1st row) and isosurface (2nd row) rendering results under different crop sizes using the ionization (He→He+) data set at time step 80. The chosen isovalue is $v = -0.9$.

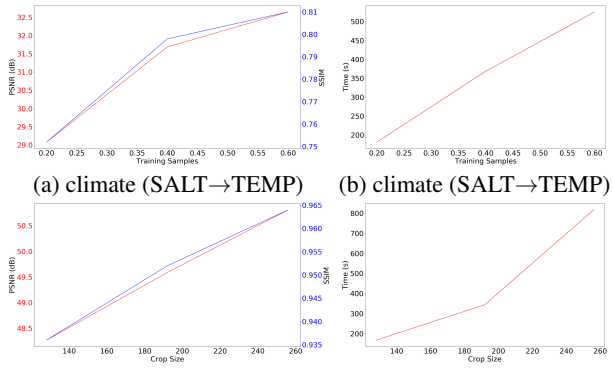


(a) V2V w/o FTM (b) V2V (c) GT

Fig. 19: Comparison of volume rendering results under different architectures using the combustion (MF→CHI) data set at time step 70.

192×96^2, and 256×128^2 using the ionization (He→He+) data set. The average PSNR and SSIM curves are shown in Figure 17 (c). We can see that using a larger subvolume size helps as V2V can learn richer semantic information. As for visual quality, we calculate the difference images [13], which are provided at the bottom-right corner, for a clear comparison. As shown in Figure 18, we can now see visual differences more clearly, particularly at the head of the ionization. Even though it takes more time to train with a larger subvolume size, as shown in Figure 17 (d), we still recommend using the subvolume size of 256×128^2 to train the ionization (He→He+) data set.

Feature translation module. To study what influences the visual quality of the volumes generated by V2V, we conducted such an experiment that trains V2V without using the feature translation module (FTM), i.e., the three purple transformation blocks shown in Figure 2 (a). The results are shown in Figure 19. We can see that more green content and less yellow content are rendered from the volume generated by V2V without FTM. We speculate that FTM serves the role of refining and filtering the features extracted at different scales during variable translation, improving translation quality.



(a) climate (SALT→TEMP) (b) climate (SALT→TEMP)

(c) ionization (He→He+) (d) ionization (He→He+)

Fig. 17: Comparison of hyperparameter settings. (a) Average PSNR and SSIM under different training samples. (b) Average training time (per epoch) under different training samples. (c) Average PSNR and SSIM under different crop sizes. (d) Average training time (per epoch) under different crop sizes.

5 Conclusions and Future Work

We have presented V2V, a new deep learning solution for selecting variables and translating variable sequences for MTVD analysis and visualization. Leveraging GAN, V2V can map one variable sequence to another variable sequence while achieving better visual quality of direct volume rendering and isosurface rendering than HM and two other deep learning solutions (Pix2Pix and CycleGAN). Besides qualitative comparison, quantitative evaluation results using PSNR (data-level), SSIM (image-level), and IS (feature-level) also confirm the effectiveness of our approach.

V2V can work in the in situ visualization setting. In this scenario, at simulation time, we store the complete sequence for one variable (i.e., all the time steps) while saving the rest of variable sequences sparsely (i.e., only the early time steps) for storage saving. During postprocessing, these reduced variable sequences are synthesized back to their original sequences with high fidelity.

V2V is part of our research effort on *data augmentation* for scientific visualization, which refers to the addition of *spatial*, *temporal*, and *variable* details to reduced data by incorporating information derived from internal and external sources. V2V addresses the *variable*-domain data augmentation, while our previous work on TSR-TVD [13] addresses the *temporal*-domain data augmentation. We are working on SSR-TVD [12], the *spatial*-domain data augmentation, to complete this research. In the future, we would like to extend our framework to handle multiple variable translations. That is, given a variable sequence, our framework can *simultaneously* extrapolate multiple variable sequences using multi-domain translation [5]. Besides, we will also explore other applications of the extracted features, for example, utilizing these features in other scientific data generation and analysis tasks, such as super-resolution and feature tracking.

Acknowledgments

This research was supported in part by the U.S. National Science Foundation through grants IIS-1455886, CCF-1617735, CNS-1629914, DUE-1833129, and IIS-1955395, and the NVIDIA GPU Grant Program. The authors would like to thank the anonymous reviewers for their insightful comments.

REFERENCES

[1] 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.

[2] 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.

[3] C.-K. Chen, C. Wang, K.-L. Ma, and A. T. Wittenberg. Static correlation visualization for large time-varying volume data. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 27–34, 2011.

[4] Z. Chen, W. Zeng, Z. Yang, L. Yu, C.-W. Yu, M. Raj, and H. Qu. LassoNet: Deep lasso-selection of 3D point clouds. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):195–204, 2020.

[5] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo. StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[6] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2016.

[7] R. Girdhar, D. F. Fouhey, M. Rodriguez, and A. Gupta. Learning a predictable and generative vector representation for objects. In *Proceedings of European Conference on Computer Vision*, pages 484–499, 2016.

[8] M. Glatter, C. Mollenhour, J. Huang, and J. Gao. Scalable data servers for large multivariate volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1291–1299, 2006.

[9] L. Guo, S. Ye, J. Han, H. Zheng, H. Gao, D. Z. Chen, J.-X. Wang, and C. Wang. SSR-VFD: Spatial super-resolution for vector field data analysis and visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 71–80, 2020.

[10] 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*, 26(4):1732–1744, 2020.

[11] 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.

[12] J. Han and C. Wang. SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2020. Under Minor Revision.

[13] J. Han and C. Wang. TSR-TVD: Temporal super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):205–215, 2020.

[14] 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.

[15] 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.

[16] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. G. Nashed, and T. Peterka. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):23–33, 2020.

[17] F. Hong, C. Liu, and X. Yuan. DNN-VolVis: Interactive volume visualization supported by deep neural network. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 282–291, 2019.

[18] 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.

[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] 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.

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

[22] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. 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.

[23] X. Liang, S. Di, D. Tao, Z. Chen, and F. Cappello. An efficient transformation scheme for lossy data compression with point-wise relative error bound. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 179–189, 2018.

[24] X. Liang, S. Di, D. Tao, S. Li, S. Li, H. Guo, Z. Chen, and F. Cappello. Error-controlled lossy compression optimized for high compression ratios of scientific datasets. In *Proceedings of IEEE International Conference on Big Data*, pages 438–447, 2018.

[25] M. Lin, Q. Chen, and S. Yan. Network in network. In *Proceedings of International Conference for Learning Representations*, 2014.

[26] X. Liu and H.-W. Shen. Association analysis for visual exploration of multivariate scientific data sets. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):955–964, 2016.

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

[28] 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.

[29] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Proceedings of Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

[31] W. P. Porter, Y. Xing, B. R. von Ohlen, J. Han, and C. Wang. A deep learning approach to selecting representative time steps for time-varying multivariate data. In *Proceedings of IEEE Conference on Visualization (Short Papers)*, pages 131–135, 2019.

[32] H. Qu, W.-Y. Chan, A. Xu, K.-L. Chung, K.-H. Lau, and P. Guo. Visual analysis of the air pollution problem in Hong Kong. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1408–1415, 2007.

[33] 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.

[34] 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.

[35] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching-incorporating a global constraint into MRFs. In *Proceedings of IEEE International Conference on Computer Vision*, pages 993–1000, 2006.

[36] 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.

[37] 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.

[38] J. Sukharev, C. Wang, K.-L. Ma, and A. T. Wittenberg. Correlation study of time-varying multivariate climate data sets. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 161–168, 2009.

[39] 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.

[40] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

[41] 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.

[42] S. Weiss, M. Chu, N. Thuerey, and R. Westermann. Volumetric isosurface rendering with deep learning-based super-resolution. *IEEE Transactions on Visualization and Computer Graphics*, 2019. Accepted.

[43] Y. Xie, E. Franz, M. Chu, and N. Thuerey. tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. *ACM Transac-*

tions on Graphics, 37(4):95:1–95:15, 2018.

[44] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *Proceedings of European Conference on Computer Vision*, pages 649–666, 2016.

[45] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics*, 9(4):119:1–119:11, 2017.

[46] 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.

[47] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of IEEE International Conference on Computer Vision*, pages 2223–2232, 2017.

APPENDIX

1 CHOICE OF FEATURE EXTRACTOR

For the choice of feature extractor, we tried different network architectures, such as autoencoder, GAN, and U-Net. All these approaches generate similar features in the projection view. However, U-Net achieves better reconstruction quality than autoencoder and requires less training time than GAN. Hence, we chose U-Net as our feature extractor. The U-Net parameter details are reported in Table 1.

Table 1: U-Net architecture parameter details.

type	kernel size	output channels	output size
input	N/A	1	$L \times H \times W$
Conv+ReLU	4	64	$L/2 \times H/2 \times W/2$
Conv+ReLU	4	128	$L/4 \times H/4 \times W/8$
Conv+ReLU	4	256	$L/8 \times H/8 \times W/8$
Conv+ReLU	$L/8 \times H/8 \times W/8$	512	$1 \times 1 \times 1$
DeConv+ReLU	4	256	$L/8 \times H/8 \times W/8$
Conv+ReLU	3	256	$L/8 \times H/8 \times W/8$
DeConv+ReLU	4	128	$L/4 \times H/4 \times W/4$
Conv+ReLU	3	128	$L/4 \times H/4 \times W/4$
DeConv+ReLU	4	64	$L/2 \times H/2 \times W/2$
Conv+ReLU	3	64	$L/2 \times H/2 \times W/2$
DeConv+ReLU	4	32	$L \times H \times W$
Conv+Tanh	3	1	$L \times H \times W$

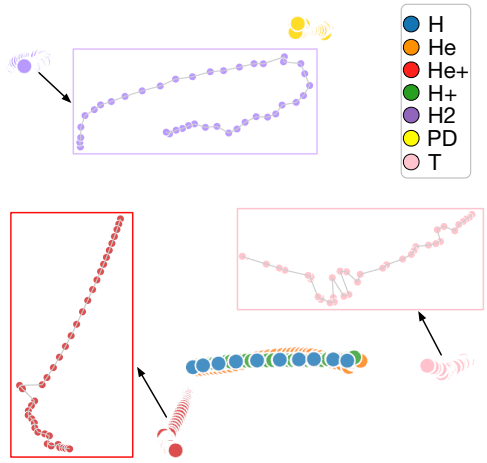


Fig. 1: t-SNE projection of the ionization data set. Each point in the projection space represents one time step. The zoomed-in views show how the time steps vary for the selected variables.

2 FEATURE UTILIZATION

Besides applying extracted features for measuring the similarity between different variables, we can leverage them for useful analysis of the multivariate time sequence, such as highlighting which time steps are shifted or which variables should be prioritized. As shown in Figure 1, we show how the ionization data set varies as the time step goes in each variable sequence. In the figure, each point in the t-SNE projection represents one time step. We connect these points from the first to the last. As we can observe, the pattern exhibited by the features from one variable can highlight which time steps have shifted or deviated from the rest to guide the selection of representative time steps. In addition, if multiple variables are displayed in the feature space, we can also find the similarities and differences among those variables. For instance, based on the distances in the projection space for the ionization data set, we can see that H2, PD, and T should be simulated anyway (i.e., we cannot rely on V2V to synthesize these variable sequences), while H, H+, He, and He+ are transferable. Therefore, scientists can save the whole sequence of H and only the early sequences of H+, He, and He+ for V2V training and translation.

3 CHOICES OF BASELINES

In general, the translation solution can be classified into two categories: *direct translation* and *disentangled translation*. Direct translation, such as Pix2Pix and CycleGAN, establishes a neural network to learn a mapping function from input to output. It can handle large data since cropping is supported. Disentangled translation, such as [2, 1, 3], first disentangles the input into two independent vectors (e.g., *content* and *style* vectors), and then swaps different content and style vectors from different inputs to generate diverse outputs. However, this approach could not handle large data since the content and style vectors are *global* information that cannot be extracted from cropped data. In addition, the data generated by disentangled translation are "fake" (i.e., there is no corresponding ground truth), which leads to a difficulty in evaluation. As such, we chose direct translation solutions (Pix2Pix and CycleGAN) as our variable-to-variable translation baselines.

4 CHALLENGES FOR IN SITU APPLICATION

It is possible to apply V2V to in situ settings. Our in situ context is formulated as follows. In multivariate time-varying data simulations, scientists first generate all variable sequences from the early time steps. Once those data are collected, scientists can leverage our variable selection solution to determine the transferable variable pairs and decide which variables may not need to be simulated further. The simulation then resumes, with only a subset of variables being simulated and their data stored. This scheme can reduce the cost of the simulation and save disk storage. During post hoc analysis, scientists can apply the trained V2V model to generate the missing variable sequences for further analysis and visualization.

Nevertheless, applying such a solution to the in situ context poses several challenges. First, as the network training could only be done offline (as the training takes a long time), the simulation needs to be "paused" once the training data are obtained. Halting the simulation entails additional work, such as saving the checkpoints. Second, we assume that the network trained using the early time steps would serve as a reasonably good predictor for inferencing variable sequences at the later time steps, which may depend on each application's nature and may not always hold. Third, for image translation, a neural network model could be trained on a large amount of data to generate diverse and realistic images. However, this is not the case for scientific data because of the lack of training data and the added training cost (2D images vs. 3D volumes). However, one can still train a model on a certain domain (e.g., combustion) with limited training data and later apply the model to translate different variable sequences or ensemble runs of the same domain. To conclude, although feasible, incorporating V2V into in situ settings requires significant research effort.

REFERENCES

[1] H. Y. Lee, H. Y. Tseng, J. B. Huang, M. Singh, and M. H. Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of European Conference on Computer Vision*, pages 35–51, 2018.

[2] M. Y. Liu, T. Breuel, and J. Kautz. Unsupervised image-to-image translation networks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[3] T. Park, J. Y. Zhu, O. Wang, J. Lu, E. Shechtman, A. A. Efros, and R. Zhang. Swapping autoencoder for deep image manipulation. *arXiv preprint arXiv:2007.00653*, 2020.