SSR-VFD: Spatial Super-Resolution for Vector Field Data Analysis and Visualization

Li Guo* Nankai University
Shaojie Ye† Zhejiang University
Jun Han‡ University of Notre Dame
Hao Zheng§ University of Notre Dame
Han Gao¶ University of Notre Dame
Danny Z. Chen‖ University of Notre Dame
Jian-Xun Wang** University of Notre Dame
Chaoli Wang†† University of Notre Dame

2020 IEEE Pacific Visualization Symposium (PacificVis) 978-1-7281-5697-2/20/$31.00 ©2020 IEEE 10.1109/PacificVis48177.2020.8737

ABSTRACT

We present SSR-VFD, a novel deep learning framework that produces coherent spatial super-resolution (SSR) of three-dimensional vector field data (VFD). SSR-VFD is the first work that advocates a machine learning approach to generate high-resolution vector fields from low-resolution ones. The core of SSR-VFD lies in the use of three separate neural nets that take the three components of a low-resolution vector field as input and jointly output a synthesized high-resolution vector field. To capture spatial coherence, we take into account magnitude and angle losses in network optimization. Our method can work in the in situ scenario where VFD are downsampled at simulation time for storage saving and these reduced VFD are upsampled back to their original resolution during postprocessing. To demonstrate the effectiveness of SSR-VFD, we show quantitative and qualitative results with several vector field data sets of different characteristics and compare our method against volume upscaling using bicubic interpolation, and two solutions based on CNN and GAN, respectively.

Index Terms: Spatial super-resolution, vector field data, convolutional neural network, deep learning.

1 INTRODUCTION

In many scientific applications, scientists run large-scale scientific simulations and generate high-quality vector field data (VFD) for the investigation of flow behaviors in various physical and natural phenomena. However, they could only afford to store a fraction of simulation output for post hoc analysis since in exascale computing, the rate of data production is much higher than available I/O bandwidths. In this paper, we focus on generating the *spatial super-resolution* (SSR) of these VFD. That means, given a low-resolution vector, for example, of size $64 \times 64 \times 64$, we aim to generate the corresponding high-resolution vector field, for example, of size $256 \times 256 \times 256$ or even $512 \times 512 \times 512$.

Generating high-resolution vector fields from low-resolution ones is meaningful for scientific applications due to the following reasons. First, many digital imaging devices still produce low-resolution vector data sets. These data sets can benefit from upscaling techniques for more effective data exploration and better examination of features of interest. Second, similar to the common practice of data compression then decompression in the scientific data analysis pipeline, SSR also fits the pipeline well by downsampling VFD at simulation time and upscaling them back during postprocessing. SSR can outperform state-of-the-art compression algorithms by more faithfully preserving the details under the same compression rate. Third, scientific simulations often simulate a large number of runs with different parameter settings but could only afford to store a small number of runs. If we could downsample individual vector fields in the first place assuming that effective upscaling can be achieved, then scientists can afford to save more runs to enable more accurate investigation of the dynamic nature of the simulation.

Upscaling a low-resolution vector field to a high-resolution one poses three key challenges. First, unlike images where each of the *rgb* channels keeps the same value range, VFD could have dramatically different value ranges for each of the *uvw* components (e.g., $u \in [-1,1]$, $v \in [-0.1,0.7]$, and $w \in [-10^{-6},10^{-4}]$). Directly concatenating these components and sending them into *convolutional neural networks* (CNNs) or *generative adversarial networks* (GANs) may literally eliminate the component with the small range as large components dominate small ones in the convolutional operation. Second, we need to consider both *global* and *local* information simultaneously during upscaling. Conventional approaches usually employ simple trilinear interpolation or *bicubic interpolation* (BI) for vector data upscaling. These interpolations are only based on local information around the interpolated position, and therefore, may blur features and miss details in streamline rendering. Achieving the desired quality is challenging if we only consider local information during upscaling. Third, VFD are sensitive in terms of both magnitude and angle. Only directly leveraging *mean squared error* (MSE) into loss design cannot capture the dynamic angle changes. On the other hand, just measuring angle difference could lead to unexpected flow pattern since angle difference does not capture magnitude difference. To synthesize high-quality vectors, we must consider both magnitude and angle differences.

To respond, we propose a deep learning solution which learns, non-uniformly and non-locally, the relationships between low-resolution vector fields and high-resolution ones. The low-resolution vector fields could be obtained by downsampling the original volumes using the bicubic kernel with a factor of 4 or even 8 in each dimension. During inference, our solution can upsample the low-resolution VFD back to the original resolution. Inspired by image and volume super-resolution techniques, we propose SSR-VFD for generating SSR from VFD. SSR-VFD consists of three separate neural nets to produce spatially coherent SSR. We train SSR-VFD by minimizing the loss function in terms of vector magnitude and angle losses. To demonstrate the effectiveness of our approach, we show quantitative and qualitative results with several data sets of different characteristics. We compare SSR-VFD against the widely-used BI, a solution based on CNN, and a solution based on GAN. We show that SSR-VFD can achieve better quality in terms of *peak signal-to-noise ratio* (PSNR) and *average angle difference* (AAD).

Our contributions are as follows. First, our work is the first that applies deep learning for generating SSR of VFD. Second, for loss function design, previous work only considers MSE loss or percep-

*e-mail:gl_1997@outlook.com, equal contribution
†e-mail: jaye@zju.edu.cn, equal contribution
‡e-mail: jhan5@nd.edu
§e-mail: hzheng3@nd.edu
¶e-mail: hgao1@nd.edu
‖e-mail: dchen@nd.edu
**e-mail: jwang33@nd.edu
††e-mail: chaoli.wang@nd.edu

2020 IEEE Pacific Visualization Symposium (PacificVis)
978-1-7281-5697-2/20/$31.00 ©2020 IEEE

tual loss while our method takes into account both magnitude and angle differences. Third, we propose a new architecture for vector field super-resolution task, which is different from the architectures commonly used in image and volume super-resolution tasks. Fourth, we investigate several hyperparameter settings and analyze how they could impact the performance of SSR-VFD.

2 Related Work

Deep learning for scientific visualization. With the explosive growth of modern deep learning techniques, researchers have recently started to explore the capabilities of deep neural nets to address various scientific visualization problems.

For volume visualization, Zhou et al. [25] presented a CNN-based approach for volume upscaling that can preserve more structural details than bicubic upscaling. Han and Wang [7] presented TSR-TVD, a recurrent generative network for interpolating immediate volume sequence given a pair of volumes from a time-varying data set. Weiss et al. [21] presented an image-space solution that learns to upsample a sampled representation of geometric properties of an isosurface at low resolution to a higher resolution. Other deep learning works have also been done to depict and explore complex volumetric structures [2], to design a generative model for volume rendering [1], to synthesize high-resolution and perceptually authentic images [11], and to estimate viewpoint quality [18]. He et al. [10] designed InSituNet that collects training data from ensemble simulations in situ, performs offline training, and enables interactive post hoc exploration and analysis.

For flow visualization, Hong et al. [12] used *long short-term memory* (LSTM) to predict data access patterns in particle tracing in order to hide the I/O latency in distributed and parallel flow visualization. Xie et al. [23] proposed tempoGAN to synthesize spatial super-resolution volume sequences where temporal coherence is guaranteed through wrapping velocity and/or vorticity fields into the synthesized volumes. Han et al. [5] designed an autoencoder to learn the latent features of flow lines or surfaces and utilized dimensionality reduction for interactive clustering and representative selection. Wiewel et al. [22] took a LSTM-based approach to predict dense 3D+time functions of physics system using an autoencoder. Kim and Günther [13] combined filter and feature extraction using CNN for robust reference frame extraction from unsteady 2D vector fields. Han et al. [6] proposed a two-stage machine learning method for vector field reconstruction that takes the streamlines traced from the original vector fields as input and outputs reconstructed high-quality vector fields.

Although generating a spatial high-resolution volumetric scalar field from a low-resolution one has been studied [23, 25], to our best knowledge, in the venue of scientific visualization, no work has been done that generates a spatial high-resolution vector field from a low-resolution one, which is the focus of this work.

Image super-resolution. Deep learning has achieved impressive results in image super-resolution tasks. Dong et al. [4] used bicubic interpolations to upsample a low-resolution image and trained a network with three convolutional layers to generate high-resolution and high-quality images. Ledig et al. [15] proposed a GAN optimized by adversarial and perceptual losses to infer photorealistic natural images for a scaling factor of 4 along each dimension. Li et al. [16] leveraged a feedback forward network that builds the feedback connections between low-level and high-level information to generate high-resolution images. Zhang et al. [24] designed an end-to-end deep model which enriches high-resolution details by adaptively transferring the texture from reference images. Soh et al. [19] presented a CNN to reconstructing realistic super-resolved images while maintaining the naturalness of the results.

Our work differs from the above works. First, we take into account the different value ranges in the three vector components. Second, unlike image super-resolution tasks where content loss and perceptual loss are considered, we design magnitude loss and angle loss to encourage the generation of high-quality vectors.

3 SSR-VFD

Let us denote $\mathbf{F}^{\mathrm{L}} = \{\mathbf{F}_1^{\mathrm{L}}, \cdots, \mathbf{F}_n^{\mathrm{L}}\}$ as a set of low-resolution vector fields where $\mathbf{F}_i^{\mathrm{L}}$ is the ith low-resolution vector field, and $\mathbf{F}^{\mathrm{H}} = \{\mathbf{F}_1^{\mathrm{H}}, \cdots, \mathbf{F}_n^{\mathrm{H}}\}$ as a set of high-resolution vector fields where $\mathbf{F}_i^{\mathrm{H}}$ is the high-resolution counterpart of $\mathbf{F}_i^{\mathrm{L}}$. Each $\mathbf{F}_i^{\mathrm{L(H)}}$ can be decomposed into three velocity components, $\mathbf{F}_{u,i}^{\mathrm{L(H)}}$, $\mathbf{F}_{v,i}^{\mathrm{L(H)}}$, and $\mathbf{F}_{w,i}^{\mathrm{L(H)}}$. Assume L, H, and W are the dimensions of high-resolution vector fields $\mathbf{F}^{\mathrm{H}}$, respectively, given a scaling factor f, L/f, H/f, and W/f are the dimensions of low-resolution vector fields $\mathbf{F}^{\mathrm{L}}$, respectively. θ_U,θ_V, and θ_W are the learnable parameters of three separate networks (i.e., u-Net, v-Net, and w-Net), respectively.

We aim to estimate a mapping function $\mathscr{S}$ from low-resolution vector fields $\mathbf{F}^{\mathrm{L}}$ to high-resolution vector fields $\mathbf{F}^{\mathrm{H}}$. Namely, $\mathbf{F}^{\mathrm{H}} = \mathscr{S}(\mathbf{F}^{\mathrm{L}})$. As shown in Figure 1, SSR-VFD accepts $\mathbf{F}^{\mathrm{L}}$ as input and decomposes $\mathbf{F}^{\mathrm{L}}$ into three components: $\mathbf{F}_u^{\mathrm{L}}$, $\mathbf{F}_v^{\mathrm{L}}$, and $\mathbf{F}_w^{\mathrm{L}}$. These three components are processed by three independent neural nets: u-Net, v-Net, and w-Net, and $\hat{\mathbf{F}}_u^{\mathrm{H}}$, $\hat{\mathbf{F}}_v^{\mathrm{H}}$, and $\hat{\mathbf{F}}_w^{\mathrm{H}}$ are synthesized by the three neural nets, respectively. Finally, SSR-VFD concatenates $\hat{\mathbf{F}}_u^{\mathrm{H}}$, $\hat{\mathbf{F}}_v^{\mathrm{H}}$, and $\hat{\mathbf{F}}_w^{\mathrm{H}}$ into $\hat{\mathbf{F}}^{\mathrm{H}}$. To minimize the difference between $\hat{\mathbf{F}}^{\mathrm{H}}$ and $\mathbf{F}^{\mathrm{H}}$, we consider both magnitude and angle losses. The computed difference will be propagated to u-Net, v-Net, and w-Net for searching the globally optimized solutions of θ_U, θ_V, and θ_W. In this section, we first describe the details of SSR-VFD, including the definition of the loss function and the architecture of SSR-VFD. Then, we provide optimization details for training SSR-VFD.

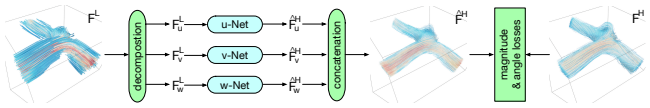


Figure 1: Overview of SSR-VFD. During training, a low-resolution vector field $\mathbf{F}^{\mathrm{L}}$ is decomposed into three components. Each component is separately processed by one neural net. Finally, the synthesized high-resolution components are concatenated to generate the high-resolution vector field $\hat{\mathbf{F}}^{\mathrm{H}}$. Magnitude and angle losses are computed for optimizing SSR-VFD.

3.1 Loss Function

Intuitively, we can compute the mean squared error (MSE) between the generated high-resolution vector field and the ground truth to optimize the network parameters by backpropagation. However, in VFD, measuring vector quality should consider two equally important attributes: *speed* and *direction*. Specifically, both attributes have great influence in streamline tracing and errors could accumulate, leading to inaccurate streamline tracing and rendering results. Thus, we propose magnitude loss and angle loss to govern the training of SSR-VFD.

Magnitude loss. We define the magnitude loss as

$$\mathscr{L}_M = \frac{1}{3 \times k \times L \times H \times W} \sum_{j=1}^{k} \sum_{i \in \{u,v,w\}} ||\mathbf{F}_{i,j} - \hat{\mathbf{F}}_{i,j}||_2, \qquad (1)$$

where k is the number of training samples, $||\cdot||$ is L_2 norm, and $\mathbf{F}_{i,j}$ and $\hat{\mathbf{F}}_{i,j}$ are the ground truth vector component value and the synthesized vector component value at the jth training sample and the ith vector component, respectively.

Angle loss. We define the angle loss as

$$\mathscr{L}_A = \frac{1}{k \times L \times H \times W} \sum_{j=1}^{k} \arccos(\mathbf{F}_j, \hat{\mathbf{F}}_j), \qquad (2)$$

where k is the number of training samples, $\arccos(\cdot)$ is the inverse cosine function, and $\mathbf{F}_j$ and $\hat{\mathbf{F}}_j$ are the ground truth vector and the synthesized vector at the jth training sample, respectively.

Taking both losses into consideration, the final loss function is defined as

$$\mathscr{L} = \lambda \mathscr{L}_A + (1-\lambda)\mathscr{L}_M, \tag{3}$$

where $\lambda \in [0,1]$ controls the relative importance of angle loss.

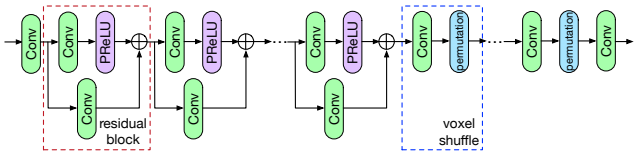


Figure 2: Network architecture of SSR-VFD. Each neural net consists of two Conv layers, K RBs, and M VS layers.

3.2 Architecture

As shown in Figure 1, SSR-VFD takes a low-resolution vector field $\mathbf{F}^{\mathrm{L}}$ as input, utilizes three separate neural nets to process the three vector components individually, and jointly outputs a synthesized high-resolution vector field $\hat{\mathbf{F}}^{\mathrm{H}}$. To enhance the performance, we leverage three techniques in SSR-VFD: (1) adding *residual blocks* (RBs) [9] to prevent the gradient vanishing problem, (2) applying the *parametric rectified linear unit* (PReLU) [8] as an activation function, and (3) leveraging *voxel shuffle* (VS) layers [7] for upscaling. The RBs bridge the feature maps from earlier layers to later layers so that the gradient can be computed through multiple paths, which alleviates the gradient vanishing problem during training. In an RB, the input is convoluted with one convolutional (Conv) layer without changing the spatial size, and the other Conv layer is applied to the input. These two outputs are connected by skip connection. Unlike ReLU, where there is no learnable parameter, PReLU allows SSR-VFD to adaptively filter the feature maps in different Conv layers. Instead of utilizing deconvolutional layers for upscaling where checkerboard artifacts will be introduced, we apply VS layers, which can speed up the training while reducing artifacts. Adding these three techniques effectively alleviates gradient vanishing, allows us to establish deep networks, and accelerates the training.

A RB accepts a feature map as input, operates a series of Convs to refine the feature map, and outputs a new feature map which is added to the input feature map. Such an example is highlighted with the red dashed box in Figure 2. For voxel shuffle, we are given a feature map with size $[C,L,H,W]$, where C is the number of channels, L,H,W are the dimensions of the feature map, and an upsampling factor u. It applies one Conv to generate a feature map with $[u^3C,L,H,W]$, then permutes the feature map into a feature map with $[C,uL,uH,uW]$. Such an example is highlighted in the blue dashed box in Figure 2.

In general, there are two common architectures for super-resolution tasks: *post-upsampling* and *pre-upsampling* [20]. The difference between these two architectures lies in where Conv layers are applied. In post-upsampling, Conv is applied *before* upscaling the inputs while in pre-upsampling, Conv is utilized *after* upscaling the inputs. In this paper, we choose the post-upsampling architecture due to the following reasons. First, pre-sampling requires higher computational cost (in terms of both space and time) since all Conv operations are performed in a high-dimensional space. However, in post-sampling, the feature extraction process through the most costly nonlinear Conv layers only occurs in a low-dimensional space and the resolution increases only at the very end of the network. As a result, the computational cost is significantly reduced, and it also leads to considerably faster training and inference. Second, in pre-sampling, BI is leveraged for upscaling low-resolution inputs to high-resolution ones, which could introduce some side effects (e.g., noise amplification and blurring), which cannot be completely cleaned via Conv layers (as the role of Conv layers is for refining BI results). In post-sampling, the upscaling process is completed through Conv layers which can reduce these side effects.

The core of SSR-VFD lies in three separate neural nets. As shown in Figure 2, each net contains two Conv layers, K RBs, and M VS layers. Each RB has two submodules: one consists of one Conv layer followed by PReLU, and the other one contains one Conv layer. These two submodules are bridged by skip connection. We set the kernel size to $3\times3\times3$, padding with 1 in all Conv layers in all RBs. Specifically, we leverage one Conv layer to extract the feature maps from the low-resolution vector field, then use K RBs for refining the features, followed by M VS layers for upscaling the features learned at low resolution to high-resolution. Finally, another Conv layer is applied. Note that M is determined by the scaling factor f (i.e., $M = \lfloor \log_2 f \rfloor$). No activation function is applied at the final Conv layer. The parameter details are listed in Table 1.

Table 1: SSR-VFD architecture parameter details for one neural net.

type	kernel size	output channels	output size
input	N/A	1	$L/f \times H/f \times W/f$
Conv+PReLU	9	32	$L/f \times H/f \times W/f$
$K\times$RB	3	32	$L/f \times H/f \times W/f$
$M\times$VS	3	32	$L \times H \times W$
Conv	5	1	$L \times H \times W$

3.3 Optimization

The process of training SSR-VFD is as follows. Given the training data, namely, k pairs of vector fields $(\mathbf{F}^{\mathrm{L}}, \mathbf{F}^{\mathrm{H}})$ and initialized network parameters θ_U, θ_V, and θ_W, we update SSR-VFD iteratively using stochastic gradient descent (SGD) until the required number of epochs is reached. At each epoch, SSR-VFD goes through all the training sample pairs. For each pair, given a low-resolution vector field $\mathbf{F}_i^{\mathrm{L}}$, SSR-VFD outputs a synthesized high-resolution one $\hat{\mathbf{F}}_i^{\mathrm{H}}$. Based on Equation 3, the gradients $\nabla_{\theta_U}\mathscr{L}$, $\nabla_{\theta_V}\mathscr{L}$, and $\nabla_{\theta_W}\mathscr{L}$ are computed, respectively. The parameters θ_U, θ_V, and θ_W can be updated through the computed gradients and the optimizer using the predefined learning rate. During inference, we run SSR-VFD in the same manner as training with the exception that gradient is not computed.

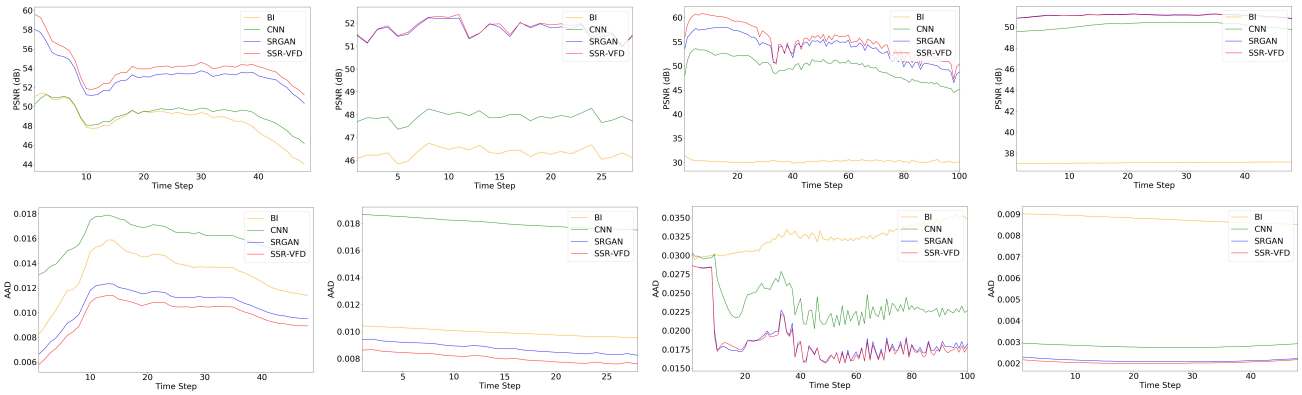
Table 2: The dimensions and training epochs of each data set.

data set	high-res dimension ($x\times y\times z\times n$)	epochs
hurricane	$500\times500\times100\times48$	1,250
solar plume	$252\times252\times1024\times28$	2,000
square cylinder	$192\times64\times48\times100$	500
supernova	$256\times256\times256\times120$	1,500
tornado	$128\times128\times128\times50$	500
vessel	$280\times260\times180\times600$	500

4 Results and Discussion

4.1 Data Sets and Network Training

Table 2 shows the simulation data sets we experimented with. In all except one case, the number of samples (n) are different time steps of the same data set. The exception is for the vessel data set, where different samples come from different model runs of steady hemodynamic simulations with varying model parameter settings. A single NVIDIA TESLA P100 GPU was used for training. We obtained the low-resolution vector fields by downsampling the high-resolution ones using the bicubic kernel. Note that we can apply SSR-VFD to vector fields of arbitrary size as it is fully convolutional. For optimization, we initialized parameters in SSR-VFD using those suggested by He et al. [8] and applied the Adam optimizer [14] to update the parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$). We set one training



(a) hurricane (b) solar plume (c) square cylinder (d) tornado

Figure 3: Comparison of PSNR (top row) and AAD (bottom row) of synthesized vector fields using BI, CNN, SRGAN, and SSR-VFD.

sample per mini-batch and the learning rate to 10^{-4}. All these hyperparameters are determined based on experiments. We randomly sampled 50% data for training and the rest is used for inference.

4.2 Results

Baselines. We use three baselines for comparison with SSR-VFD:

- BI: Bicubic interpolation (BI) is a common method for upscaling volumetric data [23, 25].
- CNN [4]: CNN contains two BI layers for upscaling and three Conv layers for refining. We also utilize magnitude and angle losses to train this model.
- SRGAN [15]: We transform SRGAN originally designed for 2D image super-resolution for 3D vector field super-resolution. In the original implementation, the generator has 16 RBs and perceptual loss is applied (which is computed based on the VGG19 ImageNet classifier). However, in 3D vector fields, more GPU memory is required and no such a classifier is offered. Therefore, we use 5 RBs and replace use perceptual loss with magnitude and angle losses.

Note that we cannot fine-tune CNN and SRGAN based on pre-trained image super-resolution models. Image super-resolution solutions are based on 2D convolution, which is difficult to fine-tune and directly apply to 3D volumetric data (i.e., processing 3D data requires 3D convolution). Therefore, we train CNN and SRGAN from scratch.

We point out that all streamline visualization results presented in the paper are synthesized by SSR-VFD are the inferred results (i.e., the network does not see these vector fields during training). All streamline rendering results for the same data set use the same setting (i.e., the same set of randomly placed seeds and the same viewing parameters). We trace 200 streamlines for the solar plume data set and 500 streamlines for all other data sets. In reference to the ground truth (GT), we compare SSR-VFD results against those generated by BI, CNN, and SRGAN.

Evaluation metrics. We utilize PSNR to evaluate the quality of synthesized vector fields. PSNR is defined as

$$\text{PSNR}(\mathbf{F},\hat{\mathbf{F}}) = 20\log_{10} I(\mathbf{F}) - 10\log_{10}\text{MSE}(\mathbf{F},\hat{\mathbf{F}}), \quad (4)$$

where $\mathbf{F}$ and $\hat{\mathbf{F}}$ are, respectively, the original and synthesized vector fields, $I(\mathbf{F})$ is the difference between the maximum and minimum values of $\mathbf{F}$ and $\text{MSE}(\mathbf{F},\hat{\mathbf{F}})$ is the MSE between $\mathbf{F}$ and $\hat{\mathbf{F}}$.

We also apply AAD to evaluate the quality of synthesized vector fields [5]. AAD is defined as

$$\text{AAD}(\mathbf{F},\hat{\mathbf{F}}) = \frac{1}{L\times H\times W}\arccos(\mathbf{F},\hat{\mathbf{F}})/\pi. \quad (5)$$

For visualizing the error between the synthesized and GT vector fields, we define the error e_j at the jth voxel as

$$e_j(\mathbf{F},\hat{\mathbf{F}}) = \sqrt{\sum_{i\in u,v,w} ||\mathbf{F}_{i,j} - \hat{\mathbf{F}}_{i,j}||_2}. \quad (6)$$

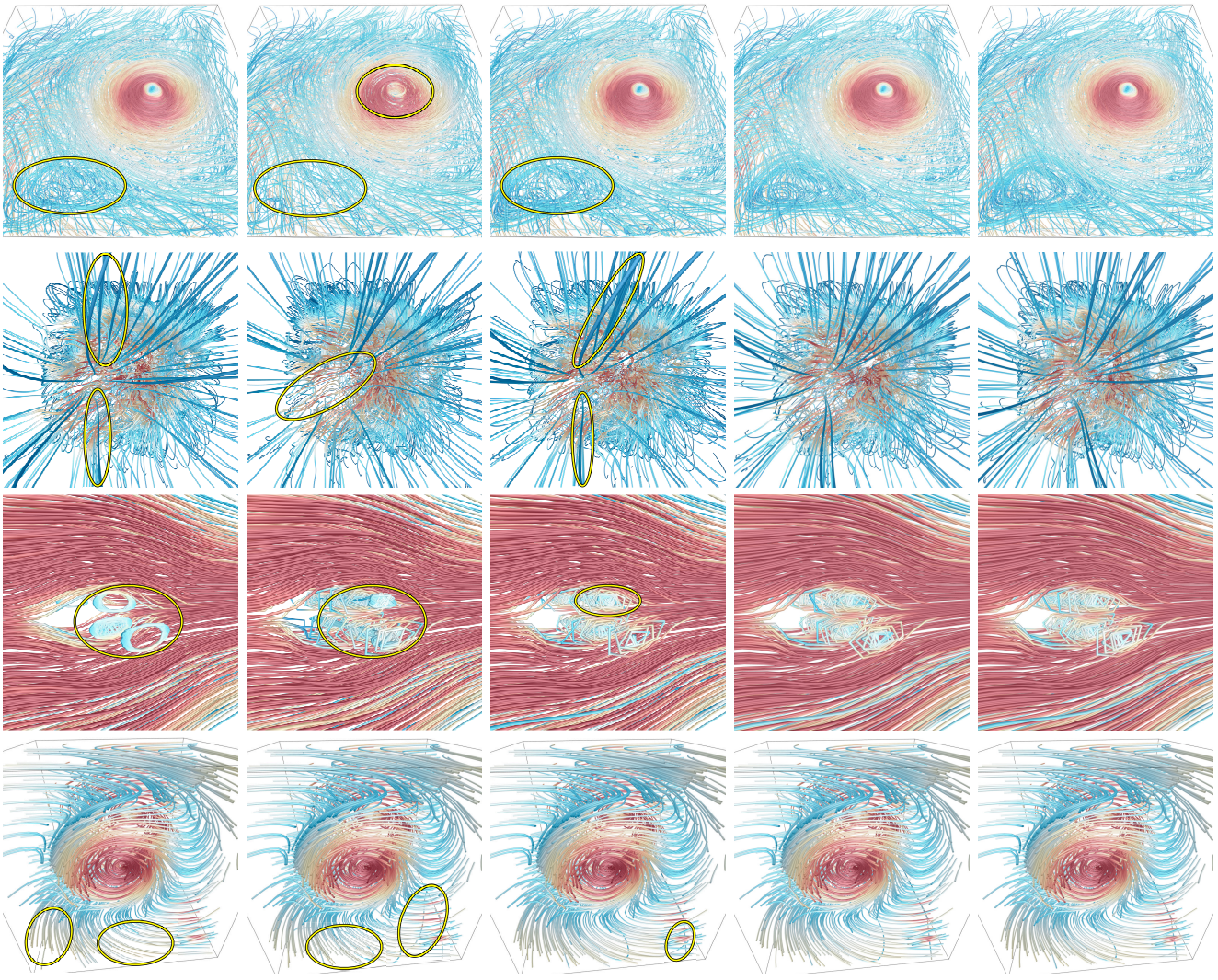
Table 3: Average PSNR and AAD values with a scaling factor of 4. The best ones are highlighted in bold.

data set	method	PSNR (dB)	AAD
hurricane	BI	49.45	0.013
	CNN	49.91	0.016
	SRGAN	53.65	0.0112
	SSR-VFD	**54.49**	**0.010**
solar plume	BI	44.25	0.0134
	CNN	46.24	0.00197
	SRGAN	48.89	0.0099
	SSR-VFD	**48.93**	**0.0093**
square cylinder	BI	30.27	0.0324
	CNN	49.68	0.0236
	SRGAN	53.25	0.0187
	SSR-VFD	**54.19**	**0.0186**
tornado	BI	38.30	0.087
	CNN	50.32	0.0028
	SRGAN	**51.15**	0.0021
	SSR-VFD	51.11	**0.0020**

Table 4: Comparison of average training time per epoch (in second) and average inference time (in second) using different methods.

data set	method	training	inference
hurricane	CNN	20.89	6.85
	SRGAN	120.09	7.62
	SSR-VFD	49.67	7.62
square cylinder	CNN	15.78	0.18
	SRGAN	84.84	0.22
	SSR-VFD	34.73	0.22
tornado	CNN	21.14	0.58
	SRGAN	113.52	0.66
	SSR-VFD	46.85	0.66
vessel	CNN	16.15	5.47
	SRGAN	90.97	6.24
	SSR-VFD	38.28	6.24

Quantitative and qualitative analysis. In Figure 3, we quantitatively compare SSR-VFD results against those generated by BI, CNN, and SRGAN using PSNR (higher is better) and AAD (lower



(a) BI (b) CNN (c) SRGAN (d) SSR-VFD (e) GT

Figure 4: Comparison of streamline rendering results. Results given by CNN, SRGAN, and SSR-VFD are the inferred results (i.e., the networks do not see these vector fields during training). Top to bottom: hurricane, solar plume, square cylinder, and tornado. We highlight differences with respect to the ground truth in yellow ellipses.

is better). We can see that in general, SSR-VFD achieves better performance compared with BI, CNN, and SRGAN. In terms of PSNR, SSR-VFD produces the highest PSNR values for the hurricane, solar plume, and square cylinder data sets while SRGAN achieves the highest PSNR values for the tornado data sets. For the tornado data set, all methods show a flat pattern since the changes among different vector fields are small. For the hurricane data set, SSR-VFD achieves the best PSNR among all methods. There is a drop in the PSNR curves at the 10th vector field since the flow pattern changes near the hurricane's eye. For the square cylinder data set, it is obvious that SSR-VFD produces the highest PSNR values compared with BI, CNN, and SRGAN.

In terms of AAD, SSR-VFD can still generate lower AAD compared with BI, CNN, and SRGAN. For the tornado data set, SRGAN and SSR-VFD yield very close AAD curves, and both are better than BI and CNN. For the hurricane data set, SSR-VFD outperforms BI, CNN, and SRGAN. It is a clear winner for the solar plume and square cylinder data sets: SSR-VFD produces an average AAD of 0.0093 and 0.0186, but BI only produces an average AAD of 0.0134 and 0.0324, respectively. However, for the square cylinder data set, the AAD curves exhibit a sudden decrease at the 15th vector fields for CNN, SRGAN, and SSR-VFD. This is because the flow pattern changes from two swirls to one swirl. In Table 3, we report the average PSNR and AAD values over the entire vector field for BI, CNN, SRGAN, and SSR-VFD. Again, SSR-VFD performs the best in terms of PSNR and AAD in all except one case (where SRGAN performs the best in terms of PSNR for the tornado data set).

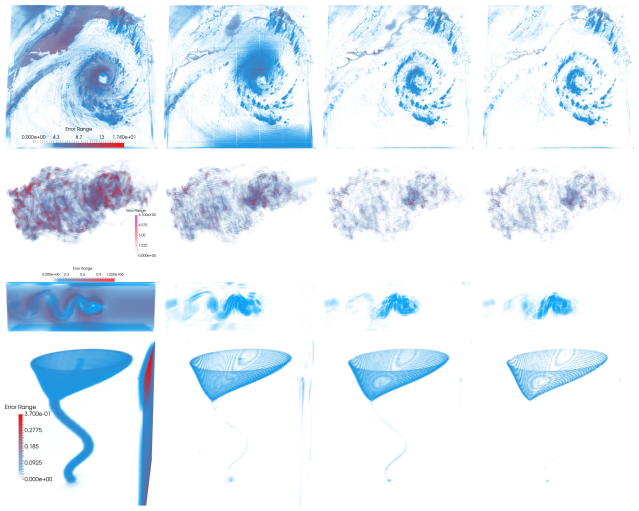
In addition, we report the average training time per epoch and average inference time using different data sets, as shown in Table 4. In terms of training time, we can observe that CNN takes the shortest time compared with SRGAN and SSR-VFD since CNN is a shallow network with three Conv layers while SRGAN requires the longest time since SRGAN consists of two networks. The entire training times for the hurricane, square cylinder, tornado, and vessel data sets are 17, 4.8, 6.5, and 5.3 hours, respectively. The model size of SSR-VFD is 9.7 MB. In terms of inference time, all three methods do not exhibit a significant difference. Therefore, SSR-VFD achieves a good trade-off between speed and performance.

In Figure 4, we compare streamline rendering results of the synthesized vector fields generated by BI, CNN, SRGAN, and SSR-

VFD. For the hurricane data set, SSR-VFD can preserve the flow patterns better near the hurricane's eye and at the bottom-left region compared with BI, CNN, and SRGAN. For the solar plume data set, BI, CNN, and SRGAN fail to trace the streamlines at the central and top-left regions, However, SSR-VFD can generate these details better. For the square cylinder data set, it is clear that SSR-VFD generates a better visual result. SSR-VFD can produce more details at the central region (i.e., the two swirls), while BI and CNN fail to recover the two swirls, and SRGAN can recover the bottom swirl well but it still fails to recover the top swirl. For the tornado data set, SSR-VFD and SRGAN can capture more details compared with BI and CNN. For example, BI and CNN fail to produce the flow details at the bottom region, while SSR-VFD can trace these streamlines well such as the streamlines at the bottom-right region.

In Figure 5, we compare volume rendering results of errors introduced by the synthesized vector fields generated by BI, CNN, SRGAN, and SSR-VFD. These volumes are computed based on Equation 6. For the hurricane data set, it is obvious that SSR-VFD and SRGAN introduce fewer errors around the hurricane's eye compared with BI and CNN. For the solar plume and square cylinder data sets, both SRGAN and SSR-VFD can generate fewer errors in the plume's head and the central region of the square cylinder compared with BI and CNN. For the tornado data set, SSR-VFD clearly produces few errors at the core and boundary compared with BI, CNN, and SRGAN.

In Figure 6, we compare the rendering results of streamlines traced from the synthesized vector fields generated by SSR-VFD and the vector field compressed then decompressed using a state-of-the-art lossy compression (LC) scheme [17] (we choose this scheme as it can effectively control data distortion while significantly reducing data size). For a fair comparison, we keep the same compression rate (i.e., 64) for both methods. As we can see, the streamlines generated by LC cannot faithfully capture the flow pattern at the central region.

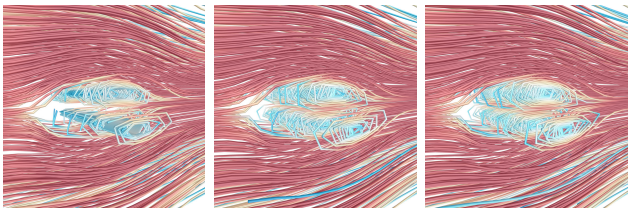


(a) BI (b) CNN (c) SRGAN (d) SSR-VFD

Figure 5: Comparison of volume rendering results of errors introduced by the synthesized vector fields. Top to bottom: hurricane, solar plume, square cylinder, and tornado.

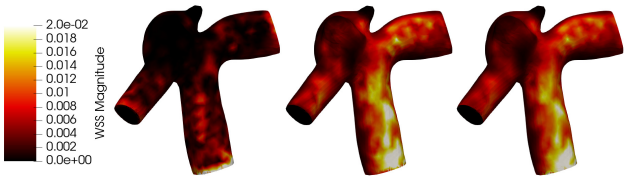
Table 5: Comparison of RMSE of velocity, vorticity, and WSS using the vessel data set. The better ones are highlighted in bold.

	velocity	vorticity	WSS
BI	0.4111	0.8484	0.8616
SSR-VFD	**0.0864**	**0.3197**	**0.2155**



(a) LC (b) SSR-VFD (c) GT

Figure 6: Comparison of streamline rendering results with SSR-VFD and LC using the square cylinder data set.



(a) BI (b) SSR-VFD (c) GT

Figure 7: Comparison of WSS of the vessel data set.

Table 6: Average PSNR and AAD under different λ values using the square cylinder data set.

λ	PSNR (dB)	AAD
0	54.31	0.0284
10^{-5}	54.28	0.0229
10^{-4}	51.32	0.0207
10^{-3}	54.19	0.0186
10^{-2}	54.08	0.0178
10^{-1}	48.54	0.0185
5×10^{-1}	47.26	0.0184
1	-10.70	0.0190

Expert evaluation of the vessel data set. We invite cardiovascular flow simulation scientists to evaluate the quality of the vessel data set. They use the measure of root-mean-square error (RMSE), which is defined as

$$\mathrm{RMSE}(\mathbf{M},\hat{\mathbf{M}}) = \sqrt{\frac{\sum_{i=1}^{N}||\mathbf{M}_i-\hat{\mathbf{M}}_i||_{L^2(\Omega_f)}}{\sum_{i=1}^{N}||\mathbf{M}_i||_{L^2(\Omega_f)}}}, \tag{7}$$

where N is the number of voxels, Ω_f denotes the computational domain, and $\mathbf{M}$ and $\hat{\mathbf{M}}$ are the GT and synthesized velocity, vorticity, or wall shear stress (WSS), respectively. Vorticity and WSS are derived based on velocity. Vorticity is defined as

$$\omega = \nabla\times\mathbf{F}, \tag{8}$$

and WSS is defined as

$$\tau_w = \left.\frac{d\mathbf{F}}{dl}\right|_{l=0}, \tag{9}$$

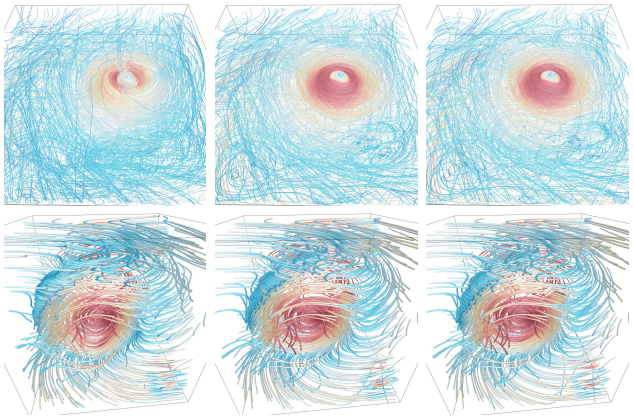
where $\mathbf{F}$ is velocity and l is the distance to the wall.

In contrast to other flow cases with regular rectangular (or cuboid) boundaries, the vessel data set represent internal flows within a complex vessel geometry, which is more challenging. In addition to velocity magnitude, the prediction performance on vorticity and WSS fields is also evaluated since these two derived hemodynamic quantities are highly related to the initialization and progression of cardiovascular diseases. Especially, the WSS is known to be the most critical hemodynamic factor to the growth and rupture of cerebral aneurysms since the endothelial cells of vascular walls are capable to

Table 7: Average PSNR and AAD under different architectures. The best ones are highlighted in bold.

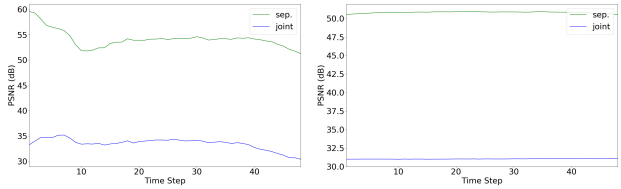
	$PSNR_u$ (dB)		$PSNR_v$ (dB)		$PSNR_w$ (dB)		PSNR (dB)		AAD	
data set	SSR w/o sep	SSR	SSR w/o sep	SSR	SSR w/o sep	SSR	SSR w/o sep	SSR	SSR w/o sep	SSR
square cylinder	49.88	**51.87**	**51.25**	50.27	25.52	**37.54**	51.93	**54.19**	0.02026	**0.01861**
tornado	49.25	**49.44**	49.24	**49.36**	54.13	**58.35**	50.85	**51.11**	0.00235	**0.00203**

sense WSS and lead to the growth remodeling of vessel structures [3]. Figure 7 shows the averaged WSS fields of all test cases by BI, SSR-VFD, and GT. It is clear that SSR-VFD produces similar WSS compared with GT. For example, at the bottom of the vessel, the WSS magnitude of SSR-VFD is close to that of GT, while BI fails to recover well the WSS magnitude. The unexpected WSS magnitude could lead to failure in identifying the most vulnerable regions of aneurysm growth and rupture. In Table 5, we compare RMSE of velocity, vorticity, and WSS between BI and SSR-VFD. Clearly, SSR-VFD leads to lower RMSE in terms of velocity, vorticity, and WSS.



(a) joint training (b) separate training (c) GT

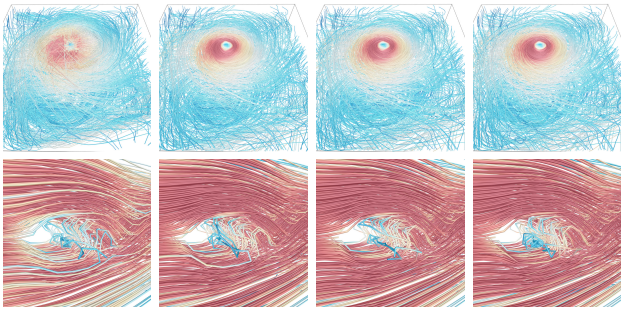
Figure 8: Different ways of training the hurricane (top row) and tornado (bottom row) data sets.



(a) hurricane (b) tornado

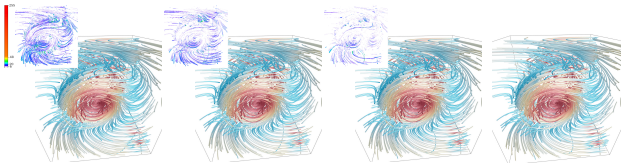
Figure 9: Comparison of PSNR using joint and separate training.

Cross-dataset evaluation. To evaluate the cross-dataset generalization of SSR-VFD, we perform joint training using the hurricane and tornado data sets. The number of epochs is the same as the one used in separate training. The streamline rendering results are shown in Figure 8. For the hurricane data set, it is clear that the streamlines generated from joint training are worse than those generated from separate training. For example, it fails to capture the flow pattern in the hurricane's eye and the top-right region. For the tornado data set, the joint model does not recover flow behavior in some regions. For example, there are few streamlines in the middle-left corner, and the streamlines around the center of the tornado are slightly sparser. We can also observe that separate training can also achieve higher PSNR, which is shown in Figure 9.



(a) $\lambda = 1$ (b) $\lambda = 0$ (c) $\lambda = 10^{-3}$ (d) GT

Figure 10: Comparison of streamline rendering results under different λ values. Top: hurricane. Bottom: square cylinder.



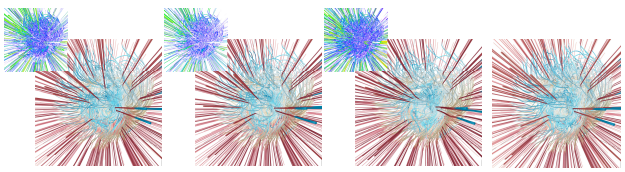
(a) 200 (b) 400 (c) 500 (d) GT

Figure 11: Comparison of streamline rendering results under different training epochs using the tornado data set. The best match with GT is the result with 500 epochs.

4.3 Hyperparameter Study

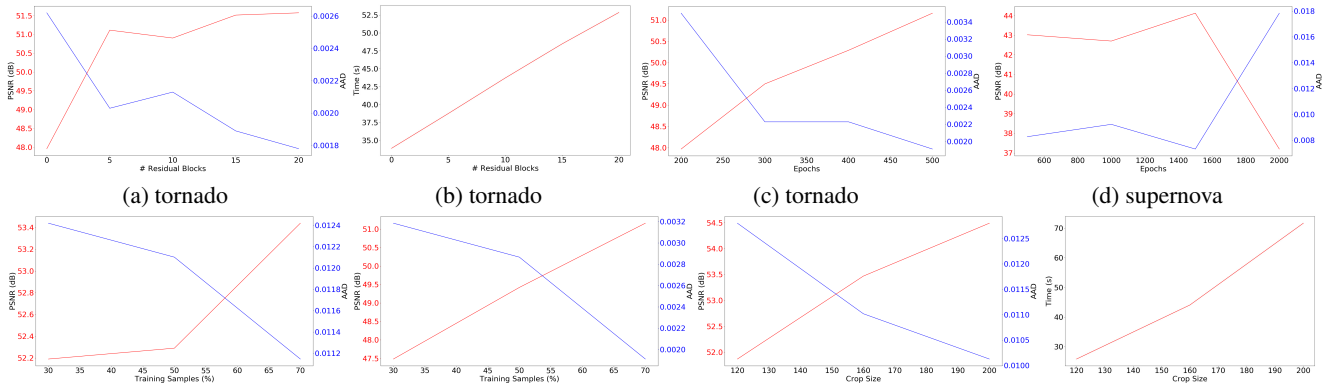
To evaluate SSR-VFD, we analyze the following hyperparameter settings: the choice of λ, the number of training epochs, the number of RBs, the number of training samples, the crop size for large vector fields, architecture design, and downsampling factor f. A detailed discussion is as follows.

λ vs. PSNR and AAD. To study the effect of λ in training, we conduct an experiment that trains SSR-VFD with different λ values. As shown in Table 6, we can observe that in general, AAD can benefit from large λ for the square cylinder data set, while PSNR can achieve the highest value when $\lambda = 0$. However, when $\lambda = 1$, we get a negative PSNR value for the square cylinder data set. This is because only considering angle loss cannot capture the magnitude error between the synthesized and GT vector fields. In Figure 10, we compare streamline rendering results using different λ values for the hurricane and tornado data sets. Both $\lambda = 0$ and $\lambda = 10^{-3}$ can



(a) 1,000 (b) 1,500 (c) 2,000 (d) GT

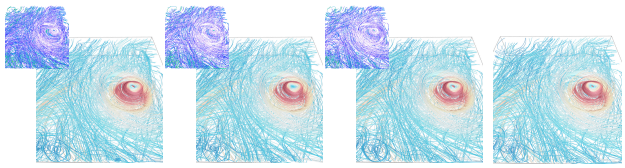
Figure 12: Comparison of streamline rendering results under different training epochs using the supernova data set. The best match with GT is the result with 1,500 epochs.



(a) tornado (b) tornado (c) tornado (d) supernova

(e) hurricane (f) tornado (g) hurricane (h) hurricane

Figure 13: Comparison of hyperparameter settings. (a) Average PSNR and AAD under different numbers of RBs. (b) Average training time (per epoch) under different numbers of RBs. (c) and (d) Average PSNR and AAD under different training epochs. (e) and (f) Average PSNR and AAD under different numbers of training samples. (g) Average PSNR and AAD under different crop sizes. (h) Average training time (per epoch) under different crop sizes.

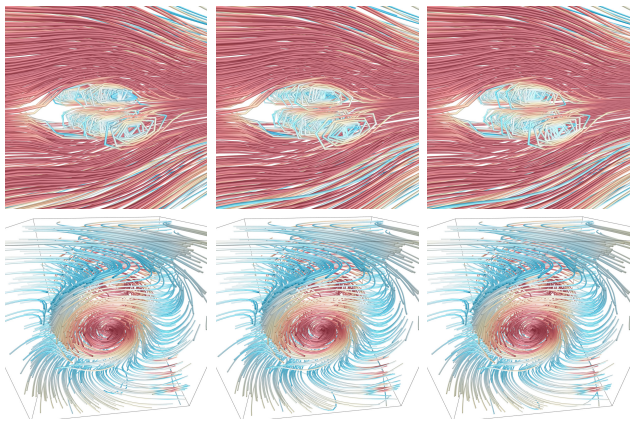


(a) $120^2 \times 100$ (b) $160^2 \times 100$ (c) $200^2 \times 100$ (d) GT

Figure 14: Comparison of streamline rendering results with different crop sizes using the hurricane data set.

capture the flow pattern well. For $\lambda = 1$, the results fail to capture the flow pattern around the hurricane's eye for the hurricane data set and fail to recover vector magnitude information for the square cylinder data set. Therefore, we choose $\lambda = 10^{-3}$ in the training to balance visual quality and PSNR and AAD values.

Training epochs vs. visual quality, PSNR, and AAD. We investigate how the quality of the synthesized vector field using SSR-VFD evolves with the increase of training epochs. Streamline rendering results after different numbers of training epochs are shown in Figures 11 and 12. For objective comparison, we calculate pixel-wise differences (the Euclidean distances) of images generated from the synthesized and original streamlines in the CIELUV color space. We map the noticeable pixel differences (with $\Delta \geq 6.0$) to nonwhite colors (clamping differences greater than 255). The difference image is displayed at the top-left corner. For the tornado data set, during the whole training process, the rendered streamlines are close to GT. For example, there is little visual difference at either the center or the bottom part of the tornado, as shown in Figure 11. We also find the average PSNR and AAD values can be improved with more training epochs, as shown in Figure 13 (c). Moreover, we observe that after 500 epochs, there is no significant difference among synthesized results with further increasing the number of epochs. Therefore, we choose 500 epochs to train the tornado data set. For the supernova data set, we find that within the first $1,500$ epochs, SSR-VFD can generate better visual quality with the increase of the number of epochs. For example, visual difference gets smaller around the center of the supernova as the training goes, as shown in Figure 12. However, after $1,500$ epochs, visual quality actually gets worse as SSR-VFD begins to overfit. The average PSNR and AAD value curves also reflect this overfitting issue, as shown in Figure 13 (d). Hence, we choose $1,500$ epochs to train the supernova data set.

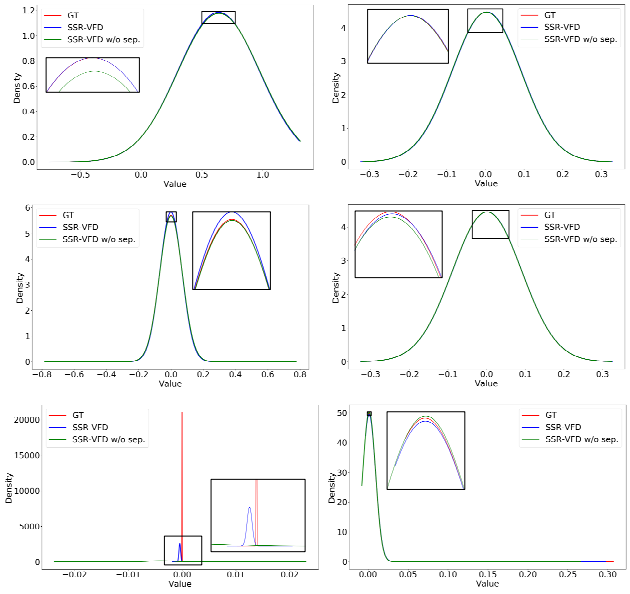


(a) SSR w/o sep (b) SSR (c) GT

Figure 15: Comparison of streamline rendering results under different architecture designs. Top: square cylinder. Bottom: tornado.

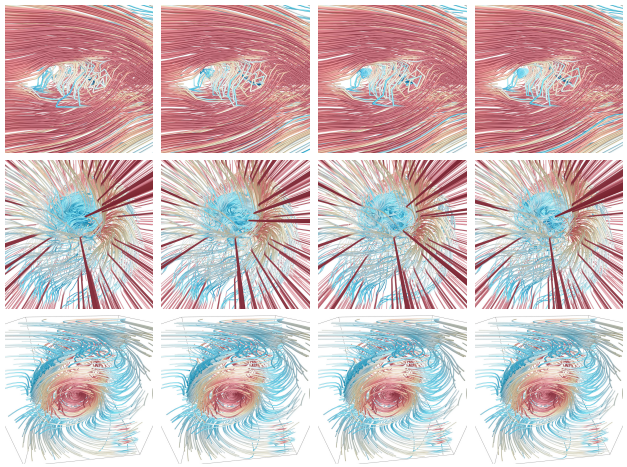
Residual blocks vs. PSNR and AAD. We investigate the influence of network depth in SSR-VFD, particularly the number of RBs on PSNR and AAD values as well as training time using the tornado data set. As shown in Figure 13 (a), for a scaling factor of 4, we report average PSNR and AAD values. As we observe, in general, more RBs can improve the average PSNR and AAD values. In addition, we observe substantial performance gain by adding more RBs. PSNR improves from 48 dB (w/o RB) to 51 dB (with 5 RBs) and the AAD decreases from 0.0026 to 0.0020. Moreover, the performance gain slowly saturates beyond 5 RBs. As for the training time, it depends approximately linearly on the number of RBs, as shown in Figure 13 (b). Therefore, we suggest applying 5 RBs in SSR-VFD as we consider this a good tradeoff between speed (including both training time and inference time) and performance.

Training samples vs. PSNR and AAD. With a scaling factor of 4, we study the influence of the number of training samples on PSNR and AAD. We use 30%, 50%, and 70% training samples to train SSR-VFD using the hurricane and tornado data sets. We plot the average PSNR and AAD curves under different numbers of training samples, as shown in Figure 13 (e) and (f). We can see that PSNR and AAD can be improved using more training samples. However, this demands a longer training time. We observe that beyond 50%, visual quality does not benefit from using more training samples. As



(a) square cylinder (b) tornado

Figure 16: Comparison of density maps of different components using the tornado (left) and square cylinder (right) data sets. Top to bottom: u, v, and w components.



(a) $f = 8$ (b) $f = 4$ (c) $f = 2$ (d) GT

Figure 17: Comparison of streamline rendering results under different scaling factors. Top to bottom: square cylinder, supernova, and tornado.

a trade-off, we suggest using 50% samples to train SSR-VFD for all data sets.

Crop size vs. visual quality, PSNR, and AAD. For large vector field data sets (i.e., the hurricane and solar plume data sets), SSR-VFD cannot afford enough memory to process the whole vector field simultaneously. Therefore, we crop subvolumes to train SSR-VFD for these data sets. We perform training with subvolume sizes of $120 \times 120 \times 100$, $160 \times 160 \times 100$, and $200 \times 200 \times 100$ using the hurricane data set. The average PSNR and AAD curves are shown in Figure 13 (g). We can observe that training SSR-VFD benefits from a larger subvolume size since an enlarged receptive field helps the network capture more semantic information. As for visual quality, we can see more visual differences in the difference images, as shown in Figure 14, particularly at the hurricane's eye and its surrounding regions. However, a larger subvolume size takes more time to train as shown in Figure 13 (h). Hence, we suggest that using a larger subvolume size to train SSR-VFD could achieve better performance.

Architecture design vs. visual quality, PSNR, and AAD. To investigate the effectiveness of SSR-VFD, we conduct an experiment that trains SSR-VFD without separating the three components using different data sets. As shown in Figure 15, we render the streamlines from the synthesized vector fields generated by these two different architectures. For the square cylinder data set, it is obvious that, compared with SSR-VFD w/o sep, SSR-VFD can better capture the features at the central region. For example, SSR-VFD w/o sep does not accurately generate the top swirl at the central region. For the tornado data set, it is clear that SSR-VFD produces high-quality streamlines since SSR-VFD w/o sep fails to recover the streamlines at the bottom-right region. In addition, in Figure 16, we plot the density maps of u, v, and w components generated by GT, SSR-VFD, and SSR-VFD w/o sep using the tornado and square cylinder data sets. For the square cylinder data set, we can observe that SSR-VFD is better than SSR-VFD w/o sep for u component while for v component, SSR-VFD w/o sep is closer to GT. For the w component, both methods fail to estimate the density map. However, SSR-VFD can still approximate a peak for w while SSR-VFD w/o sep only generates a flat line. For the tornado data set, it is clear that SSR-VFD can generate close density maps for v and w components. For example, the blue curve (SSR-VFD) is closer to the red curve (GT) compared with the green curve (SSR-VFD w/o sep) for v and w components. For a clearer comparison, we compute PSNR and AAD values for each component, as demonstrated in Table 7. As we can see, separating the vector field can bring higher PSNR values of each component for most of the data sets. In addition, the benefit diminishes for small components. For example, for the square cylinder data set, the PSNR value of w component increases to 37.54 dB from 25.52 dB due to the separation training mechanism.

Scaling factor vs. visual quality, PSNR, and AAD. To investigate the upscaling ability of SSR-VFD, we conduct an experiment that trains SSR-VFD with different scaling factors using different data sets. As shown in Figure 17, we render the streamlines from the vector fields synthesized from different scaling factors. For the square cylinder data set, it is clear that under $f = 8$, the central region cannot be recovered well, especially for the swirl at the top-left portion of the central region. However, under $f = 2$ and $f = 4$, this feature can still be preserved. For the supernova data set, there is no significant difference between the results of $f = 2$ and $f = 4$. However, for $f = 8$, we find that the color of streamlines at the central region is more bluish than GT. For the tornado data set, all results are similar to GT. But upon a close comparison, several streamlines are missing at the bottom-right region in the result with $f = 8$. Therefore, the appropriate value for f is 4 or 8 for most of the data sets we explore. Average PSNR and AAD values under different scaling factors are reported in Table 8. We can observe that SSR-VFD can produce higher PSNR and AAD values for different data sets compared to BI. In addition, the larger the scaling factor is, the more benefit SSR-VFD can bring.

5 Conclusions and Future Work

We have presented SSR-VFD, a new solution for generating spatial super-resolution of VFD. We design a neural network that takes the low-resolution vector field and its high-resolution counterpart as a pair for training. Once trained, the network is able to generate the spatially-resolved vector field given only a low-resolution vector field as input. Compared with BI, CNN, and SRGAN, SSR-VFD yields synthesized high-resolution vector fields of better visual quality, both qualitatively and quantitatively. To incorporate SSR-VFD into the in situ scenario, during preprocessing, we will train the

Table 8: Average PSNR and AAD under different scaling factors f. The best ones are highlighted in bold.

		PSNR (dB)		AAD	
data set	f	BI	SSR	BI	SSR
hurricane	2	55.67	**57.70**	**0.005**	0.007
	4	49.45	**54.49**	0.013	**0.010**
solar plume	2	57.23	**57.29**	**0.002**	0.006
	4	44.25	**48.93**	0.013	**0.009**
square cylinder	2	33.25	**62.00**	0.029	**0.018**
	4	30.27	**55.19**	0.032	**0.019**
	8	26.71	**48.60**	0.043	**0.019**
supernova	2	**48.38**	46.66	**0.008**	0.017
	4	41.56	**44.12**	**0.019**	0.023
	8	37.02	**42.42**	0.035	**0.023**
tornado	2	43.63	**53.65**	0.007	**0.001**
	4	38.30	**51.11**	0.009	**0.002**
	8	34.54	**48.53**	0.016	**0.003**

network with pre-run data. Then at simulation time, we will downsample VFD for storage saving. During postprocessing, we will upsample these reduced VFD back to their original resolution. With a recommended scaling factor of 4 or 8, we can downsample VFD by 64 or 512 times at simulation time and upsample these reduced data back to their original resolution with good quality using SSR-VFD. In the future, we would consider adding physical loss to the loss function design so that essential physical laws can be satisfied using SSR-VFD. We will also explore *temporal super-resolution* (TSR) for unsteady vector fields by taking temporal coherence into account.

Acknowledgments

This research was supported in part by the National Science Foundation (NSF) through grants IIS-1455886, CCF-1617735, CNS-1629914, and CMMI-1934300, and the Defense Advanced Research Projects Agency (DARPA) under the Physics of Artificial Intelligence (PAI) program (contract HR00111890034). Li Guo and Shaojie Ye conducted this work as iSURE (International Summer Undergraduate Research Experience) students at the University of Notre Dame during Summer 2019. 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] 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.

[3] A. Dardik, L. Chen, J. Frattini, H. Asada, F. Aziz, F. A. Kudo, and B. E. Sumpio. Differential effects of orbital and laminar shear stress on endothelial cells. *Journal of Vascular Surgery*, 41(5):869–880, 2005.

[4] 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.

[5] 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*, 2018. Accepted.

[6] 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.

[7] 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.

[8] 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*, pp. 1026–1034, 2015.

[9] 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*, pp. 770–778, 2016.

[10] 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.

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

[12] 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*, pp. 76–85, 2018.

[13] B. Kim and T. Günther. Robust reference frame extraction from unsteady 2D vector fields with convolutional neural networks. *Computer Graphics Forum*, 38(3):285–295, 2019.

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

[15] 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*, pp. 4681–4690, 2017.

[16] Z. Li, J. Yang, Z. Liu, X. Yang, G. Jeon, and W. Wu. Feedback network for image super-resolution. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3867–3876, 2019.

[17] 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*, pp. 438–447, 2018.

[18] 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.

[19] J. W. Soh, G. Y. Park, J. Jo, and N. I. Cho. Natural and realistic single image super-resolution with explicit natural manifold discrimination. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8122–8131, 2019.

[20] Z. Wang, J. Chen, and S. C. Hoi. Deep learning for image super-resolution: A survey. *arXiv preprint arXiv:1902.06068*, 2019.

[21] 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.

[22] 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.

[23] 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.

[24] Z. Zhang, Z. Wang, Z. Lin, and H. Qi. Image super-resolution by neural texture transfer. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7982–7991, 2019.

[25] 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*, pp. 38:1–38:6, 2017.