Scalar2Vec: Translating Scalar Fields to Vector Fields via Deep Learning

Pengfei Gu* Jun Han† Danny Z. Chen‡ Chaoli Wang§

University of Notre Dame

ABSTRACT

We introduce Scalar2Vec, a new deep learning solution that translates scalar fields to velocity vector fields for scientific visualization. Given multivariate or ensemble scalar field volumes and their velocity vector field counterparts, Scalar2Vec first identifies suitable variables for scalar-to-vector translation. It then leverages a k-complete bipartite translation network (kCBT-Net) to complete the translation task. kCBT-Net takes a set of sampled scalar volumes of the same variable as input, extracts their multi-scale information, and learns to synthesize the corresponding vector volumes. Ground-truth vector fields and their derived quantities are utilized for loss computation and network training. After training, Scalar2Vec can infer unseen velocity vector fields of the same data set directly from their scalar field counterparts. We demonstrate the effectiveness of Scalar2Vec with quantitative and qualitative results on multiple data sets and compare it with three other state-of-the-art deep learning methods.

Index Terms: Scalar field, vector field, scalar-to-vector translation, deep learning.

1 INTRODUCTION

To study various scientific phenomena, domain scientists often generate time-varying multivariate or ensemble volumetric data from simulations. These data can span thousands of time steps and include dozens of variables. Due to limited disk storage and I/O bandwidth, scientists could only afford to store simulation output sparsely (e.g., a fraction of scalar and vector fields) for post hoc analysis and visualization. Inspired by image colorization works (i.e., translating grayscale images to color images), we investigate the problem of *scalar-to-vector* (Scalar2Vec) *translation*, i.e., recovering velocity vector fields from their scalar field counterparts. That is, given a set of scalar volumes, we aim to translate them to the corresponding vector volumes in high quality via a deep learning approach.

Studying the Scalar2Vec problem is meaningful due to the following reasons. First, images are different from volumes, and *grayscale vs. color images* is not the same as *scalar vs. vector fields*. Therefore, Scalar2Vec itself is worth exploring, even with the success of many image colorization solutions. Second, with the trained Scalar2Vec network for a certain simulation, we could only store scalar field data of the same simulation and use them to recover their vector field counterparts during postprocessing. We will show that integrated with data compression, Scalar2Vec is superior to direct compression of vector fields, both qualitatively and quantitatively. This has direct implications for scientific workflow and data management, which is highly relevant to scientific simulation and visualization. Third, given multivariate or ensemble data, it remains unknown which scalar variable could be used in Scalar2Vec translation. Using the derived velocity magnitude (VM) scalar fields seems an obvious choice for velocity vector fields. Still, we want to design a solution that chooses different variables suitable for Scalar2Vec translation based on the similarity between VM and other variables. Note that these chosen variables are alternatives to VM, as the translation is still one-to-one (i.e., from one scalar field to the vector field). Such a solution could bring new insights for scientists to understand the complex relationship between scalar and vector fields.

Achieving Scalar2Vec translation poses several unique challenges. First, Scalar2Vec translation is a more challenging task than scalar-to-scalar translation (e.g., V2V [23]) because more information (i.e., scalars vs. vectors) and more complex patterns (e.g., swirls and spirals) need to be synthesized. This requires us to design a more powerful model for learning the relationship between scalar and vector fields. Second, investigating scalar fields of different variables other than VM for successful Scalar2Vec translation is critical to generalize this approach, while image colorization does not have such an issue. Third, multi-scale information, for example, features of different sizes or granularities, must be considered in Scalar2Vec translation, as scalar fields of different variables can vary considerably. Otherwise, the solution could lead to inferior quality because it may miss learning finer details across scales. Furthermore, achieving good translation using different scalar fields under the same network presents additional challenges, as various scalar fields could exhibit dramatically different multivariate patterns.

To address these challenges, we present Scalar2Vec, a new deep learning solution that translates scalar fields to velocity vector fields for scientific visualization. Scalar2Vec consists of two steps: *variable selection* and *field translation*. We assume that ground-truth (GT) vector fields are available during training. At the variable selection step, we utilize U-Net [44] to learn variable features in the latent space and apply t-SNE [38] to project the latent features into a 2D space. We then select the variables close to VM in the projection space as candidates for one-to-one translation. We hypothesize that it is most suitable to use VM for the translation task, and the experimental results confirm this hypothesis. At the field translation step, we propose a *k-complete bipartite translation network* (kCBT-Net) to translate the scalar fields to the corresponding velocity vector fields in high quality. The input to kCBT-Net is the sampled scalar fields of the same variable. We minimize the errors between the reconstructed vector fields and GT vector fields using a combination of magnitude, angle, and Jacobian losses. Once trained, Scalar2Vec can infer unseen velocity vector fields of the same data set directly from their scalar field counterparts. To demonstrate the effectiveness of Scalar2Vec, we provide quantitative and qualitative results on multiple data sets. We compare Scalar2Vec with three other state-of-the-art deep learning methods: Pix2Pix [31], CycleGAN [58], and V2V [23]. The experiments indicate that Scalar2Vec achieves better quantitative results in two metrics at the data level and one metric at the representation (i.e., streamline) level. Qualitative results also confirm the better visual quality of the streamlines traced from the recovered velocity vector fields using Scalar2Vec.

Our contributions are as follows. First, Scalar2Vec is the first attempt in scientific visualization that translates scalar fields to vector fields. Second, we propose a new network architecture (i.e., kCBT-Net) for Scalar2Vec, different from popular image colorization architectures. Third, we show the better performance of Scalar2Vec against Pix2Pix, CycleGAN, and V2V.

*e-mail: pgu@nd.edu
†e-mail: jhan5@nd.edu
‡e-mail: dchen@nd.edu
§e-mail: chaoli.wang@nd.edu

2 Related Work

Deep learning for scientific visualization. Researchers have tackled various scientific visualization tasks via deep learning approaches. For volume visualization, examples are super-resolution generation (single-volume super-resolution [57], temporal and spatial super-resolution [17,18], and simultaneous spatiotemporal super-resolution [22]), volume completion [21], feature learning of volumes [7] and isosurfaces [19], rendering image synthesis (volume rendering [3,26,27] and isosurface rendering [51]), viewpoint quality estimation [46], and variable to variable translation [23]. For flow visualization, examples are super-resolution generation [2,14,20,53], vector field reconstruction from field lines (for steady [16] and unsteady [12] flows), access pattern learning [28], feature learning of lines [15] and surfaces [15,19], reference frame extraction [33], and neural flow map interpolation [32]. Unlike the above works, we focus on translating scalar fields to velocity vector fields, which has not been attempted in scientific visualization. Our work is similar to V2V [23] but differs in task goal and network architecture. More importantly, our *k*CBT-Net is more powerful than the V2V architecture in capturing multi-scale information. In fact, the generator of V2V can be considered as a special case of our *k*CBT-Net.

Image-to-image translation. Researchers have achieved impressive success on image-to-image (I2I) translation with deep learning techniques. Isola et al. [31] introduced Pix2Pix, the first unified framework based on conditional GAN (cGAN) [39] for paired I2I translation. Xian et al. [52] developed TextureGAN with a local texture loss for addressing the sketch-to-image problem with texture patches. Gonzalez et al. [11] adopted disentanglement representation to improve the performance of I2I translation. Tang et al. [49] utilized the extra semantic information for cross-view image translation. The above works require paired images as input. To eliminate the dependence on paired data, Zhu et al. [58] introduced CycleGAN that uses a cycle consistency loss for unpaired I2I translation. Liu et al. [36] presented an unsupervised I2I translation framework following Coupled GANs with a shared-latent space assumption. Huang et al. [29] extended the work of Liu et al. [36] for handing multi-modal I2I translation. Park et al. [40] proposed a swapping autoencoder to encode the input image into two independent components and swap them to generate diverse outputs.

For image colorization, Cheng et al. [8] leveraged a five-layer CNN for image colorization. Zhang et al. [55] utilized CNN to colorize grayscale images. Their model consists of eight blocks, and each block has two or three stacked convolutional layers. Iizuka et al. [30] proposed a multi-path CNN to colorize the grayscale images. The presented multi-path network has two branches. Each branch is divided into four subnets to learn features at multiple paths. Zhang et al. [56] developed a user-guided image colorization solution. Their approach consists of two networks: local hint and global hint networks. Both networks are based on U-Net. He et al. [25] introduced exemplar-based local colorization via deep learning, transferring the colors from a given reference image to the grayscale one. Their method includes two sub-networks: similarity and colorization sub-networks, based on VGG19 [47] and U-Net, respectively. Ci et al. [9] presented an interactive deep cGAN for scribble-based anime line art colorization in an end-to-end style. The discriminator uses a pre-trained network called Illustration2Vec [45], and the generator adopts the U-Net architecture. Xu et al. [54] achieved fast deep exemplar colorization using a stylization-based architecture. The architecture encompasses transfer and colorization sub-networks, following the encoder-decoder and U-Net structures, respectively. Su et al. [48] proposed an instance-aware image colorization method which includes three components: an object detector for object instance detection, two instance colorization networks for feature extractions at the object- and image-levels, and a fusion module for feature blending and final color prediction.

Translating scalar fields to vector fields is a more difficult task. Grayscale and color images share the same visual content while only preserving different color channels. Vector fields exhibit dynamic flow patterns (e.g., swirling, twisting patterns, etc.), which may not present in scalar fields with distinct contents. Thus, the neural network needs to learn how to synthesize flow patterns from scalar fields and remove structures not presented in vector fields.

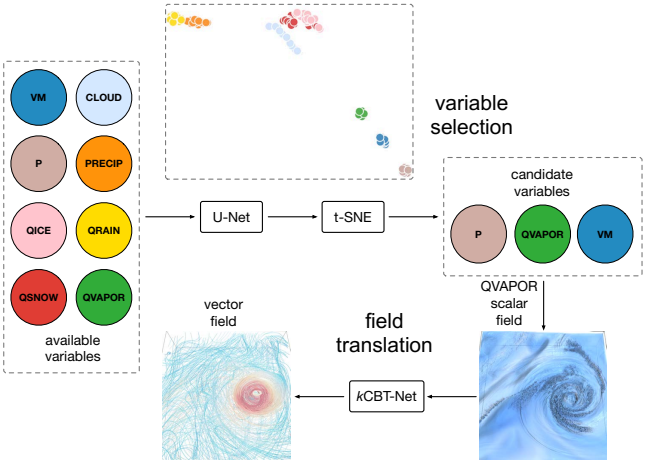


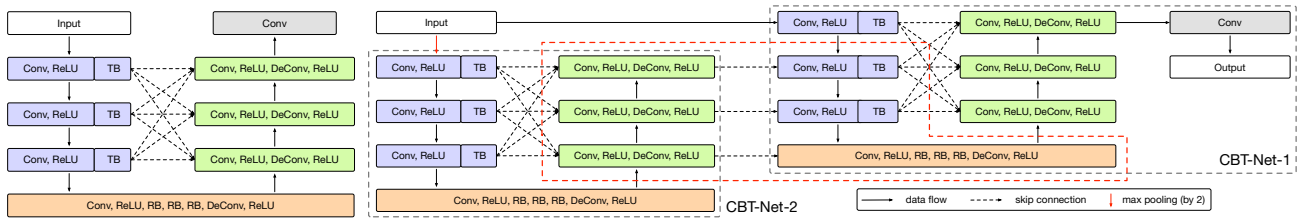
Figure 1: Overview of Scalar2Vec using the hurricane data set where three of the eight variables are chosen as candidate variables for the scalar-to-vector translation task.

3 Scalar2Vec

As shown in Figure 1, we design Scalar2Vec for accomplishing the translation task in two steps: variable selection and field translation. For variable selection, similar to V2V [23], we identify suitable variables for Scalar2Vec translation. For field translation, we propose the *k*CBT-Net to complete the translation task in high quality given the chosen variables. Scalar2Vec randomly picks volume samples based on a certain percentage threshold to select variables and train the network. Once trained, given an unseen scalar field not used in training, Scalar2Vec can infer its corresponding vector field.

3.1 Variable Selection

At the variable selection step, following V2V [23], we leverage U-Net [44] as the feature extractor to implicitly learn the features from all variables (including VM) of the given multivariate or ensemble volumetric data. In particular, the randomly sampled time steps of all variables of the given multivariate or ensemble volumetric data are input to U-Net to output the features per variable per time step. VM scalar fields are derived from their corresponding velocity vector fields. Each randomly picked volume sample of each variable is mapped to a latent feature. Then, we utilize t-SNE [38] to project all these latent features into a 2D space. As shown in Figure 1, in the projection space, each feature is mapped to a 2D point. Based on the proximity between the point set from one variable and the point set from the VM variable, we select variables that behave similarly to VM as the candidate variables. Note that the translation task is one-to-one, not many-to-one. So we choose individual candidate variables, not a subset of variables. For the example shown in Figure 1, we can observe that P and QVAPOR are the best candidates besides VM, PRECIP and QRAIN are the worst, while CLOUD, QICE, QSNOW are in the middle. After that, the sampled scalar fields of the selected variables are input to *k*CBT-Net to learn the actual Scalar2Vec translation. Note that if no other suitable variables besides VM exist, we only consider translating the scalar fields of VM to the corresponding vector fields for achieving high-quality reconstruction. Of course, one can pick arbitrary variables for Scalar2Vec translation (i.e., without variable selection stage), but this does not guarantee high-quality translation.



(a) (b)

Figure 2: Network architecture of (a) CBT-Net and (b) kCBT-Net with $k = 2$. The red dashed line delineates the module connection between CBT-Net-2 and CBT-Net-1.

3.2 Field Translation

At the field translation step, we approximate a translating function that perceives and explores *multi-scale* information. We employ three additions to ensure the high-quality Scalar2Vec translation. First, we add *residual blocks* (RBs) [24] between the encoding and decoding paths to alleviate the issue of vanishing gradients. Specifically, in one path, the input is convoluted through several convolutional (Conv) layers without changing its dimensions. In another path, the input is passed through an identity mapping. In the end, the outputs from the two paths are combined by summation. Second, we utilize a generalized *complete bipartite connections* (CBC) structure [5, 13] to consolidate feature hierarchies across scales via skip connection [44] for assimilating multi-scale information. Third, we include multiple *fully convolutional network* (FCN) [37] sub-modules to work on different scales of the input data systematically. This FCN sub-module design can further extract multi-scale information from different paths working on different scales' input.

CBT-Net. A common way to extract and reuse multi-scale information is to leverage skip connections. For example, U-Net [44] and its variants introduced skip connections to fuse the multi-scale features extracted from the encoder to the counterparts of the decoder. However, such skip connections may not guarantee full exploitation and reuse of multi-scale information.

To fully achieve the multi-scale information learning, we deploy the CBT-Net to connect between different scales (i.e., different resolutions of the feature maps) in the encoding path and different scales in the decoding path. As illustrated in Figure 2 (a), CBT-Net consists of three paths: *encoding path*, *bridge path*, and *decoding path*. There are four Conv layers in the encoding path. Each Conv layer has a stride of 2 to reduce the dimension by half. In the bridge path, we use three RBs to assist the gradient flow between the encoding and decoding paths. The decoding path has four deconvolutional (DeConv) layers followed by a Conv layer. The DeConv layers are used to upscale the dimension of the features by two. In the CBC structure, a skip connection with *transformation block* (TB) from each scale of encoding path is concatenated to the scale s ($s = 1, 2, 3$) in decoding path. We utilize TBs to translate the features from the encoding path to the decoding path. A TB consists of two paths: one path with three Conv layers, and the other with one Conv layer. The two paths are finally combined with addition. To concatenate feature maps extracted from different scales, we use trilinear interpolation for upsampling and apply max pooling for downsampling. This design encourages explicitly multi-scale feature reuse and consolidates the feature hierarchies across scales. In the model, a ReLU follows each Conv and DeConv layer. The architecture parameter details of CBT-Net are listed in Table 1. We can treat U-Net as a special case of our CBT-Net.

kCBT-Net. To further exploit multi-scale information, we adopt the multi-path diagram from Chen et al. [6]. Specifically, we connect k CBT-Net sub-modules sequentially to extract information from input at different scales. We propagate the information captured by one CBT-Net sub-module to the next CBT-Net sub-module to benefit feature extraction. That is, the information extracted by CBT-Net-t ($2 \leq t \leq k$) will be forwarded to the next sub-module CBT-Net-$(t-1)$. Figure 2 (b) gives such an example of kCBT-Net with $k = 2$. In this example, CBT-Net-2 works on the downsampled data, and CBT-Net-1 works on the data with the original resolution. kCBT-Net propagates the information extracted by CBT-Net-2 to CBT-Net-1. CBT-Net-1 takes every piece of information from CBT-Net-2 in the commensurate layers to fuse the propagated information.

Table 1: CBT-Net architecture parameter details. Note that we set the number of initial channels c to 64 for kCBT-Net.

type	kernel size	stride	output channels
input	N/A	N/A	1
Conv+ReLU	4	2	c
TB	3	1	c
Conv+ReLU	4	2	$2 \times c$
TB	3	1	$2 \times c$
Conv+ReLU	4	2	$4 \times c$
TB	3	1	$4 \times c$
Conv+ReLU	4	2	$8 \times c$
$3 \times$ RB	3	1	$8 \times c$
DeConv+ReLU	4	2	$8 \times c$
Conv+ReLU	3	1	$8 \times c$
DeConv+ReLU	4	2	$4 \times c$
Conv+ReLU	3	1	$4 \times c$
DeConv+ReLU	4	2	$2 \times c$
Conv+ReLU	3	1	$2 \times c$
DeConv+ReLU	4	2	c
Conv	3	1	3

Loss function. Similar to An et al. [2], we jointly use magnitude loss ($\mathscr{L}_m$), angle loss ($\mathscr{L}_a$), and Jacobian loss ($\mathscr{L}_j$) to optimize our model, which is formulated as

$$\mathscr{L} = \lambda_m \mathscr{L}_m + \lambda_a \mathscr{L}_a + \lambda_j \mathscr{L}_j, \tag{1}$$

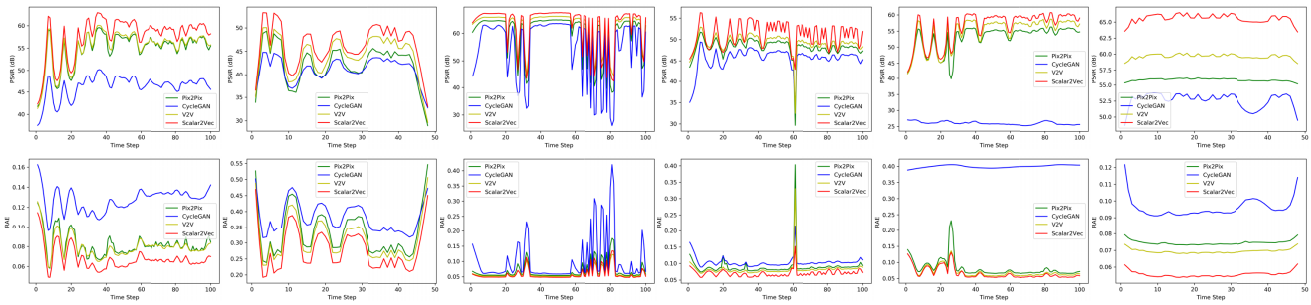
where λ_m, λ_a, and λ_j are the weights that control the relative importances of these losses. $\mathscr{L}_m$, $\mathscr{L}_a$, and $\mathscr{L}_j$ are defined as

$$\mathscr{L}_m = \sum_{i=1}^{N} ||\widehat{\mathbf{V}}_i - \mathbf{V}_i||_2, \tag{2}$$

$$\mathscr{L}_a = \sum_{i=1}^{N} (1 - \cos(\widehat{\mathbf{V}}_i, \mathbf{V}_i)), \tag{3}$$

$$\mathscr{L}_j = \sum_{i=1}^{N} ||\nabla\widehat{\mathbf{V}}_i - \nabla\mathbf{V}_i||_2, \tag{4}$$

where $\cos(,)$ is the cosine function, ∇ deotes the Jacobian function, $\widehat{\mathbf{V}}_i$ and $\mathbf{V}_i$ are the i-th training samples of the reconstructed and GT vector fields, respectively, N refers to the number of training samples, and $||\cdot||_2$ denotes the L_2 norm.



(a) half-cylinder (b) hurricane (c) supercurrent (d) supernova (e) Tangaroa (f) tornado

Figure 3: Comparison of PSNR (top row) and RAE (bottom row) of synthesized vector fields at each time step under different methods.

3.3 Optimization

The optimization of Scalar2Vec is as follows. The model takes scalar field volume samples as input and initializes the network parameters θ. Scalar2Vec is updated via stochastic gradient descent. The training continues until reaching the given number of epochs. During training, the gradients $\nabla_\theta \mathscr{L}$ are computed according to Equation (1), and the parameters θ are automatically updated through the optimizer given the specified learning rate and computed gradients. When it comes to inference, we run Scalar2Vec as usual, except that the gradients are not calculated.

Table 2: Variables and dimension of each data set.

data set	variables	dimension ($x \times y \times z \times n$)
half-cylinder [43]	VM (320), VM (640), VM (6,400)	$640 \times 240 \times 80 \times 100$
hurricane [1]	VM, CLOUD, P, PRECIP, QICE, QRAIN, QSNOW, QVAPOR	$500 \times 500 \times 100 \times 48$
supercurrent [50]	VM, RHO	$256 \times 128 \times 32 \times 100$
supernova [4]	VM	$128 \times 128 \times 128 \times 100$
Tangaroa [42]	VM	$300 \times 180 \times 120 \times 100$
tornado [10]	VM	$128 \times 128 \times 128 \times 48$

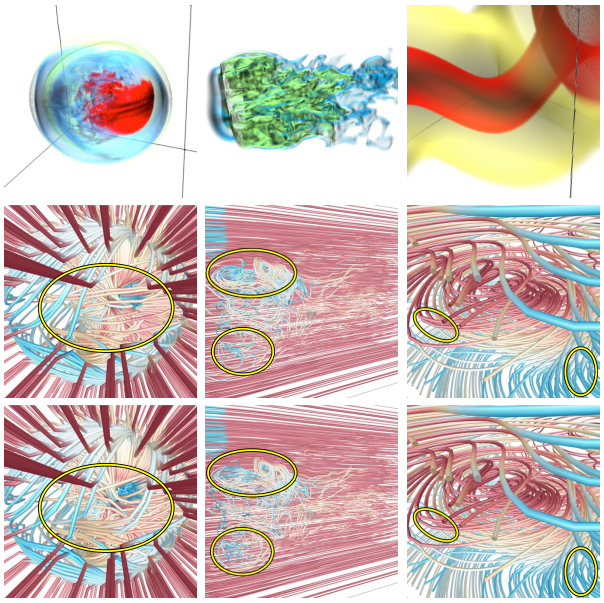
.

4 Results and Discussion

We tested the data sets shown in Table 2, where the number of volume samples (n) refers to each data set's different time steps. All VM scalar fields were derived from their corresponding velocity vector fields. Among the six data sets, the half-cylinder data set has three VM ensembles with different Reynolds numbers, the hurricane data set has eight variables, and the supercurrent data set has two variables. The rest of the data sets have only VM scalar fields.

Scalar2Vec was implemented using PyTorch [41]. We used an NVIDIA Tesla V100 graphics card with 32 GB of memory for training and inference. The network parameters were initialized using normal initialization for optimization, and the Adam optimizer [34] was used to update the parameters ($\beta_1 = 0.9, \beta_2 = 0.999$). We used one training sample for each mini-batch and trained the model with the "poly" learning rate policy. The initial learning rate was set to 10^{-4}, and the maximum number of epochs was set to $2,000$ for all data sets. For each variable used for Scalar2Vec translation, we randomly sampled 40% of volumes from the sequence for training, and the rest of 60% were used for inference. All these parameters were empirically decided based on experiments.

Note that for the ensemble half-cylinder, we trained the network on VM (640) scalar fields and used the trained network to directly infer the corresponding vector fields of VM (320), VM (640), and VM (6,400). Therefore, no training time for VM (320) and VM (6,400) is reported in Table 3. For hurricane and supercurrent, the network was trained for each given variable from scratch. The 100 time steps of the supercurrent data set was drawn evenly from the original sequence of 900 time steps.



(a) supernova (b) Tangaroa (c) tornado

Figure 4: Visualization results using VM for Scalar2Vec translation. Top to bottom: input VM scalar fields, streamlines traced from translated vector fields, streamlines traced from GT vector fields.

4.1 Results

Baselines. We compare Scalar2Vec with three baseline methods: (1) Pix2Pix [31] (the first *paired* I2I translation framework), (2) CycleGAN [58] (a deep learning method for *unpaired* I2I translation), and (3) V2V [23] (the first deep learning work in scientific visualization for variable translation). We choose Pix2Pix [31] and CycleGAN [58] from I2I translation works to compare with, for two reasons. First, these works cover both paired and unpaired I2I translation frameworks. Second, more importantly, these direct translation frameworks, unlike disentangled translation solutions (e.g., [36, 40]), can be applied to handle large data which fits our scenario. We use the original architectures (i.e., the generators) from Pix2Pix, CycleGAN, and V2V for the scalar-to-vector translation task. Note that the generator of V2V is a special case of our kCBT-Net, and our kCBT-Net is more powerful than the V2V architecture in capturing multi-scale information. For a fair comparison, we set the same number of initial, input, and output channels for all the methods.

All streamline visualization results for Scalar2Vec are produced from the inferred vector field samples (i.e., these samples are not

Table 3: Average PSNR (dB), RAE, and SD values, total training time (in days), and inference time per volume sample (in seconds) using different variables for translation. The best ones are highlighted in bold (same for the rest of the tables in the Appendix).

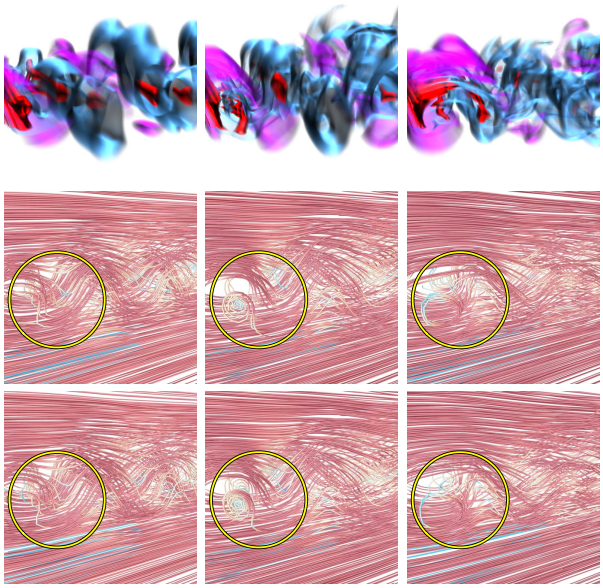
data set	method	PSNR	RAE	SD	train	infer
half-cylinder VM (320)	Pix2Pix	37.23	0.238	5.85	—	69
	CycleGAN	34.15	0.273	6.09	—	61
	V2V	37.98	0.220	5.77	—	97
	Scalar2Vec	**38.47**	**0.209**	**5.69**	—	143
half-cylinder VM (640)	Pix2Pix	54.79	0.082	0.78	3.6	68
	CycleGAN	46.43	0.127	0.82	3.3	61
	V2V	55.18	0.080	0.50	5.1	97
	Scalar2Vec	**58.61**	**0.063**	**0.40**	7.5	142
half-cylinder VM (6,400)	Pix2Pix	37.49	0.231	5.62	—	69
	CycleGAN	33.34	0.289	6.65	—	61
	V2V	38.15	0.221	5.48	—	97
	Scalar2Vec	**38.94**	**0.208**	**5.12**	—	143
hurricane (VM)	Pix2Pix	42.03	0.334	21.04	1.8	113
	CycleGAN	40.99	0.381	25.34	1.6	101
	V2V	44.03	0.313	20.82	2.6	160
	Scalar2Vec	**46.27**	**0.281**	**19.84**	3.8	236
hurricane (QRAIN)	Pix2Pix	26.85	0.864	74.33	1.9	105
	CycleGAN	24.01	1.021	98.39	1.6	94
	V2V	29.87	0.720	56.96	2.6	148
	Scalar2Vec	**36.06**	**0.513**	**43.33**	3.8	218
hurricane (QVAPOR)	Pix2Pix	38.46	0.435	31.47	1.8	106
	CycleGAN	24.00	1.022	98.05	1.6	95
	V2V	39.28	0.424	29.25	2.6	150
	Scalar2Vec	**41.83**	**0.369**	**27.62**	3.8	220
supercurrent (VM)	Pix2Pix	59.88	0.068	1.42	0.6	0.24
	CycleGAN	54.89	0.102	1.61	0.5	0.22
	V2V	62.25	0.060	1.28	0.8	0.34
	Scalar2Vec	**62.92**	**0.059**	**1.12**	1.2	0.50
supercurrent (RHO)	Pix2Pix	58.63	0.077	1.43	0.6	0.24
	CycleGAN	55.37	0.099	1.53	0.5	0.22
	V2V	62.48	0.059	1.42	0.8	0.35
	Scalar2Vec	**63.35**	**0.057**	**1.10**	1.2	0.51
supernova (VM)	Pix2Pix	48.19	0.090	0.87	1.0	0.18
	CycleGAN	45.55	0.104	0.90	0.9	0.16
	V2V	49.18	0.083	0.86	1.4	0.26
	Scalar2Vec	**51.13**	**0.071**	**0.60**	2.1	0.38
Tangaroa (VM)	Pix2Pix	52.87	0.081	0.61	3.5	25
	CycleGAN	26.07	0.399	4.32	3.1	22
	V2V	54.76	0.071	0.57	5.0	35
	kCBT-Net	**56.44**	**0.065**	**0.53**	7.3	52
tornado (VM)	Pix2Pix	55.87	0.075	0.07	0.5	0.39
	CycleGAN	52.64	0.096	0.11	0.4	0.35
	V2V	59.58	0.070	0.06	0.7	0.55
	Scalar2Vec	**65.53**	**0.055**	**0.04**	1.0	0.81

used in training). We apply the same random seed set to trace streamlines from the same vector field sample inferred from different methods. The fourth-order Runge-Kutta method is used to trace streamlines from the seed points in both forward and backward directions. Unless stated otherwise, we trace 500 streamlines to generate all streamline visualization images shown in the paper. All data sets use the same color map for streamline rendering. Streamline colors show velocity magnitudes where blue to yellow to red indicate low to medium to high velocity. Volume rendering of input scalar fields uses different color and opacity maps for different data sets.

Evaluation metrics. For quantitative evaluation, we compute peak signal-to-noise ratio (PSNR) and root-absolute-error (RAE) between the reconstructed vector fields and GT vector fields at the data level. The RAE measures the relative error between the reconstructed vector field $\widehat{\mathbf{V}}$ and GT vector field $\mathbf{V}$

$$\mathrm{RAE}(\widehat{\mathbf{V}},\mathbf{V}) = \sqrt{\frac{\sum_{i=1}^{L\times H\times W}\sum_{j\in\{u,v,w\}}||\widehat{\mathbf{V}}_{i,j}-\mathbf{V}_{i,j}||}{\sum_{i=1}^{L\times H\times W}\sum_{j\in\{u,v,w\}}||\mathbf{V}_{i,j}||}}, \qquad (5)$$

where L,H,W are the three spatial dimensions of the volume, u,v,w are the three components of each vector, and $||\cdot||$ denotes the L_1 norm. We also compute the streamline distance (SD) [12] at the representation (i.e., streamline) level. SD is computed using the mean of the closest point distances between streamlines traced from the synthesized and GT vector fields with the same seed set.



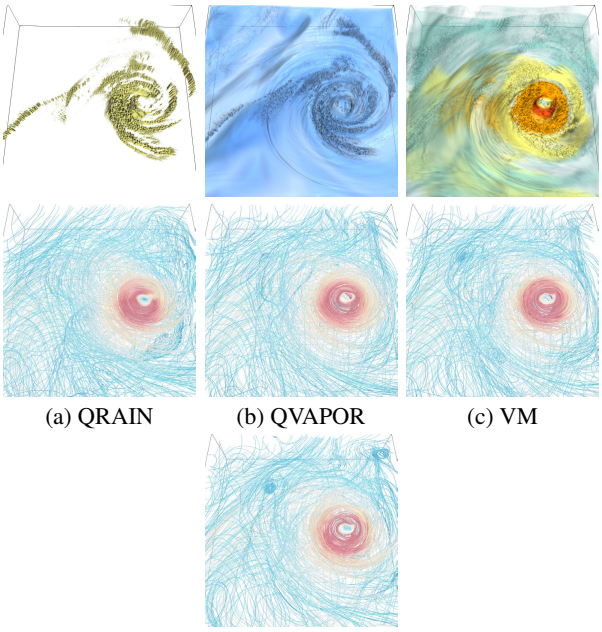
(a) VM (320) (b) VM (640) (c) VM (6,400)

Figure 5: Visualization results of the half-cylinder data set using different VM ensembles for Scalar2Vec translation. Top to bottom: input VM scalar fields, streamlines traced from translated vector fields, streamlines traced from GT vector fields.

Quantitative comparison. In Table 3, we provide quantitative comparison results of Scalar2Vec with the three baseline solutions: Pix2Pix, CycleGAN, and V2V. We compare the average PSNR and RAE values of vector fields synthesized from different scalar fields, and the SD values of their corresponding traced streamlines.

First of all, we can observe that Scalar2Vec yields the best results for all cases in terms of PSNR (highest), RAE (lowest), and SD (lowest) values, followed by V2V and Pix2Pix. CycleGAN produces the worst results. These consistent results across all data sets demonstrate the overall quantitative advantages of Scalar2Vec over other methods. For the half-cylinder data set, the results with VM (640) are clearly better than those with VM (320) and VM (6,400). This is because we use the network trained on VM (640) to directly perform scalar-to-vector translation for VM (320), VM (640), and VM (6,400). Therefore, performance degradation is expected. For the hurricane data set, it is not surprising that the results with VM are the best, followed by QVAPOR. Among the three variables, QRAIN has the worst results. It is the least suitable variable for the translation task (refer to Figure 1), as its point set is farthest away from that of VM. Among the four methods, CycleGAN performs much worst when using either QVAPOR or QRAIN. For the supercurrent data set, we can see that VM and RHO generate very similar results. RHO wins over VM across all three metrics when using CycleGAN, V2V, and Scalar2Vec, and only loses to VM when using Pix2Pix.

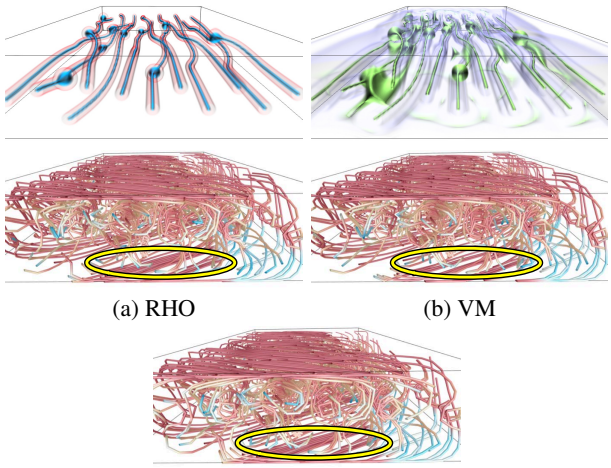
In terms of training time and inference time, we can see that Scalar2Vec takes the most time for both training and inference across all data sets, followed by V2V, Pix2Pix, and CycleGAN. No training time is reported for half-cylinder VM (320) and VM (6,400) as we train the network on VM (640) only. Scalar2Vec is the most time-consuming method due to the use of CBC and the design of k connected modules. However, the increased time cost is not too high



(a) QRAIN (b) QVAPOR (c) VM

(d) GT

Figure 6: Visualization results of the hurricane data set using different variables for Scalar2Vec translation.



(a) RHO (b) VM

(c) GT

Figure 7: Visualization results of the supercurrent data set using different variables for Scalar2Vec translation.

compared to the best baseline V2V, which is acceptable considering their vital contributions to the improvements. Note that the reported training and inference times use the model with the number of initial channels c set to 64. Nevertheless, we can save more computation cost (including training and inference times) and model storage when setting $c = 32$ without sacrificing much performance, referring to Table 3 in the Appendix.

In Figure 3, we compare PSNR and RAE of synthesized vector fields at each time step using Pix2Pix, CycleGAN, V2V, and Scalar2Vec. We can see that overall, Scalar2Vec performs the best (with the highest PSNR and lowest RAE values). For the supercurrent data set, the performance changes dramatically. This is because the flow pattern changes much among different time steps sampled from the original sequence. The changes among different time steps are small for the tornado data set. There is a sharp change in the PSNR and RAE curves for the supernova data set at the 61st time step due to the significant flow pattern changes.

Qualitative results of Scalar2Vec. In Figures 4 to 7, we show the visualization results of Scalar2Vec translation. Volume rendering results of the scalar fields are provided so that we can see how different various scalar fields are and how different the scalar and vector fields are. Streamlines traced from the recovered vector fields are compared to those generated from the GT ones.

In Figure 4, we show Scalar2Vec translation results of three data sets (supernova, Tangaroa, and tornado) using the VM scalar field as input. Streamline visualization results indicate that the overall translation quality is good. Nevertheless, for the supernova data set, Scalar2Vec does not capture well the flow pattern around the supernova's core, which changes very dramatically. The same conclusion can be drawn for the Tangaroa data set, where we can see apparent differences around the highlighted interesting flow patterns. The tornado data set's result is very good, as we can only spot slight differences for a few streamlines (referring to ellipse highlights).

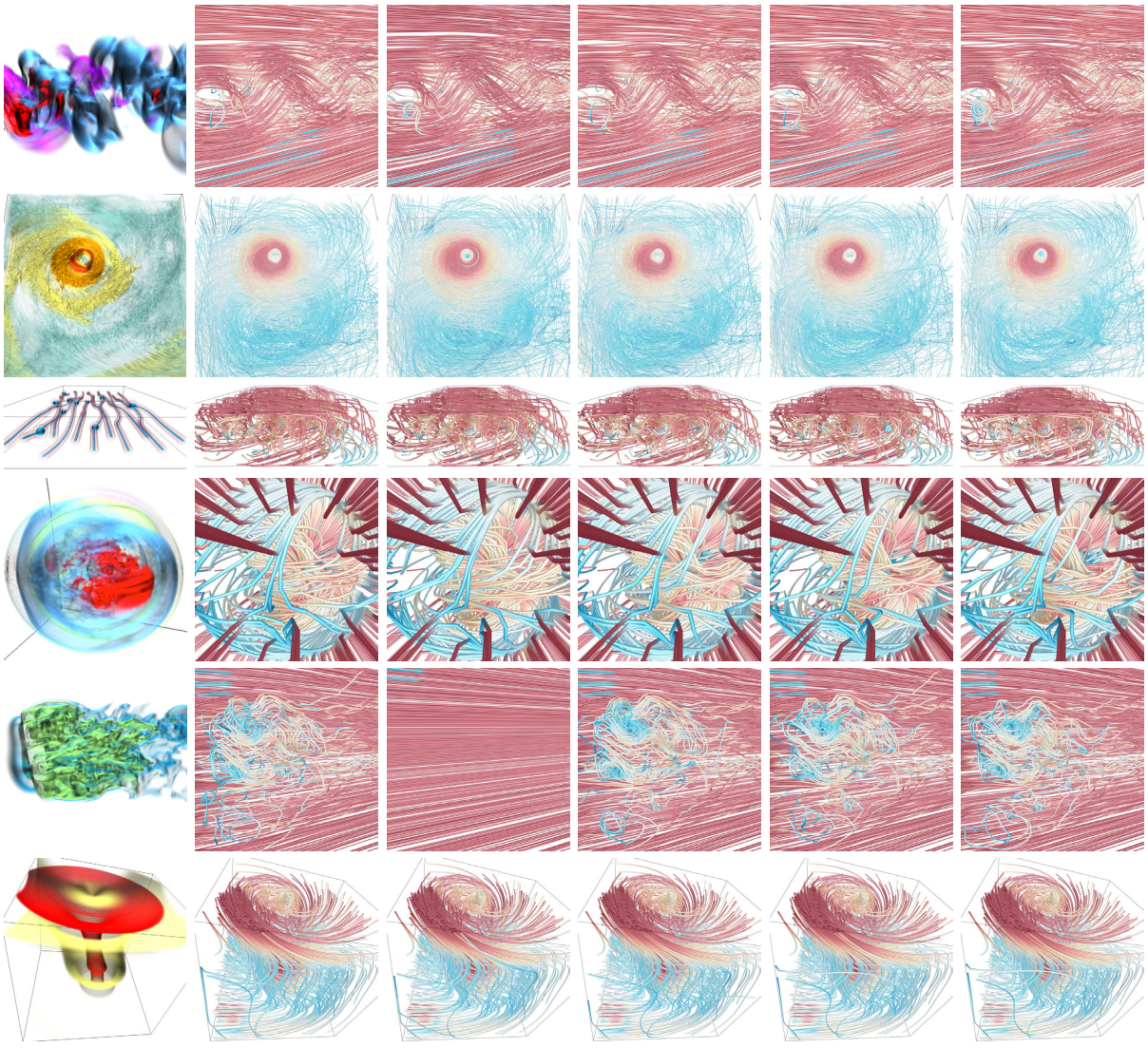
In Figure 5, we show results of the half-cylinder data set using the network trained on VM (640) for Scalar2Vec translation. It is clear that the result with VM (640) is closer to the corresponding GT result, while those with VM (320) and VM (6,400) exhibits more apparent differences with respect to their GT results. These qualitative results echo the quantitative results reported in Table 3.

In Figure 6, we show Scalar2Vec translation results of the hurricane data set using QRAIN, QVAPOR, and VM. Overall, we can observe that VM achieves the best visual quality, followed by QVAPOR. QRAIN gets the worst result. These qualitative results are consistent with the quantitative results reported in Table 3.

In Figure 7, we show results of the supercurrent data set using VM and RHO for Scalar2Vec translation. We can see that RHO generates a slightly better streamline visualization result compared to the GT, referring to ellipse highlights. These close visualization results are echoed in the quantitative results reported in Table 3 as the differences between VM and RHO are relatively small.

Qualitative comparison. In Figure 8, we compare Scalar2Vec (kCBT-Net) with Pix2Pix, CycleGAN, and V2V. We choose volume samples different from what have been shown in Figures 4 to 7 for inference. For the half-cylinder data set using VM (640), we can see that all four methods synthesize the vector field reasonably well. Taking a close examination of the "hole" in the middle-left region, Scalar2Vec performs slightly better than the other three. For the hurricane data set using VM, all four methods generate acceptable results. Still, Scalar2Vec produces the best result around the hurricane's eye (i.e., the overall shape captured by the red streamlines). For the supercurrent data set using RHO, all four methods lead to good results. Still, Scalar2Vec produces the best result at the bottom-center region. For the supernova data set using VM, again, all four methods generate similar results. Scalar2Vec outperforms the other three methods at the central region. For the Tangaroa data set using VM, we can see that CycleGAN produces the worst result as all streamlines are mostly straight. This is because the architecture of CycleGAN is simple, which cannot capture dramatically different patterns. The other three methods (Pix2Pix, V2V, and Scalar2Vec) have close results. A close comparison shows that Scalar2Vec and V2V slightly outperform Pix2Pix. These qualitative results echo the quantitative results shown in Table 3: PSNR and RAE values of CycleGAN are much worse than other methods, and Pix2Pix lags behind V2V and Scalar2Vec. For the tornado data set using VM, we can see that the overall flow is recovered very well across all four methods. Paying close attention to the two streamlines at the middle-left region, we can see that Scalar2Vec gives the best result.

In Figure 9, we select four cases from Figure 8 and show a filtered version of streamline visualization along with seeding points so that their differences can be easily observed. For each case,



(a) input (b) Pix2Pix (c) CycleGAN (d) V2V (e) *k*CBT-Net (f) GT

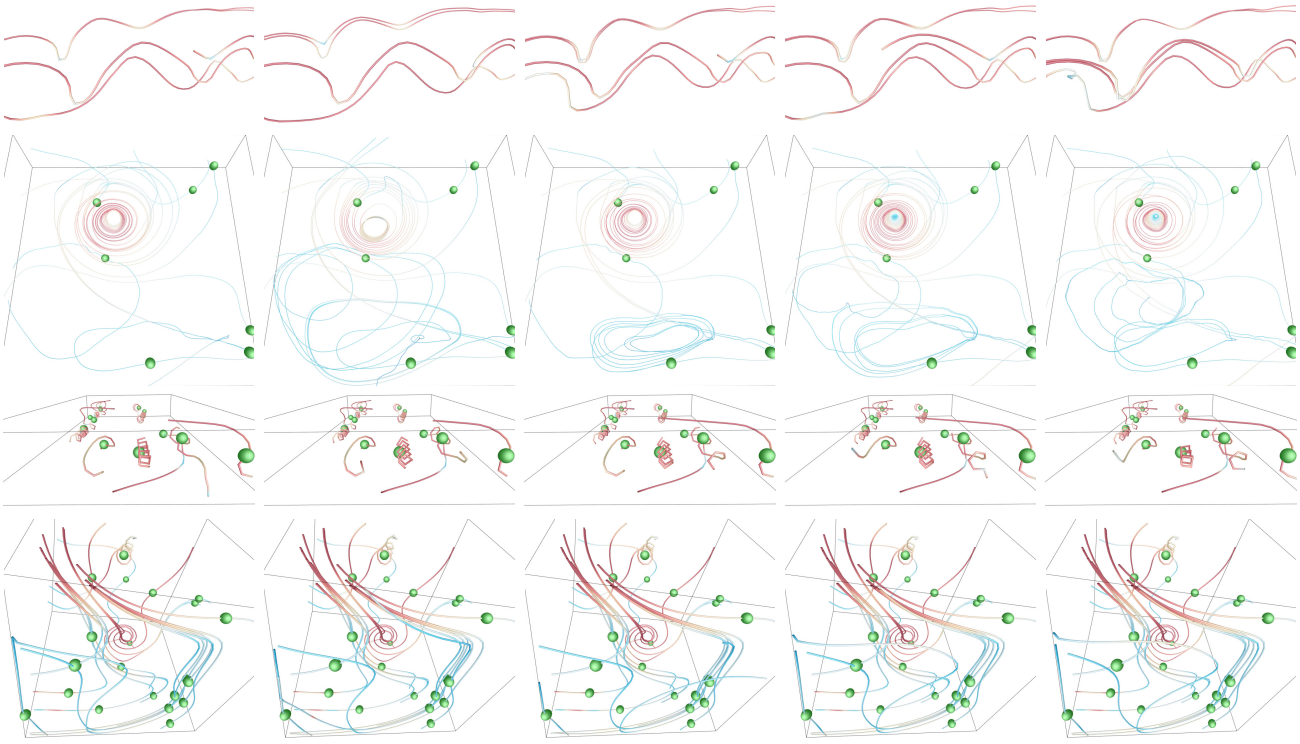
Figure 8: Streamline visualization results with different methods. Top to bottom: streamlines traced from translated vector fields with half-cylinder (VM (640)), hurricane (VM), supercurrent (RHO), supernova (VM), Tangaroa (VM), and tornado (VM) input scalar fields.

among the 500 streamlines randomly traced, we select a subset that passes through high entropy regions. The entropy is computed based on the variations of vector direction and magnitude. We gradually increase the entropy threshold to only keep a small number of streamlines for visualization clarity. The selected streamlines correspond to interesting flow features and patterns. The filtered streamline visualization results now clearly show that Scalar2Vec outperforms Pix2Pix, CycleGAN, and V2V as the shape and relative positions of the selected streamline match those of the GT the best.

In Figure 10, we compare vorticity visualization results using hurricane (VM) and supernova (VM) data sets. To better illustrate the differences, we compute pixel-wise differences (i.e., the Euclidean distances in the CIELUV color space) of volume rendering images generated from vorticity scalar fields derived from their velocity vector fields. From the difference images, we can see that for both data sets, *k*CBT-Net generates much closer results to GT than Pix2Pix, CycleGAN, and V2V.

Evaluation of data reduction. We consider a compression algorithm (i.e., SZ [35]) to further save data storage using the half-cylinder data set of different ensembles. We compare our method with SZ against directly compressing vector fields using SZ. For our method, the VM volumes are compressed using SZ under the error bound of 0.1 for data reduction and later translated to their corresponding vector fields. The data reduction rate, which is defined as the ratio between the size of original vector fields and the sum of the compressed VMs (320, 640, 6400), 40% of compressed VM (640) vector fields, and the network model, is 68.08. The data reduction rate for direct SZ compression is set the same for a fair comparison. The PSNRs (RAEs) of translated vector fields (320), (640), and (6400) are 35.85 (0.268), 39.71 (0.228), and 35.83 (0.270), respectively, and the PSNRs (RAEs) of SZ compressed vector fields (320), (640), and (6400) are 35.18 (0.309), 34.99 (0.314), and 35.07 (0.315), respectively. In Figure 11, we compare streamline visualization results with SZ and *k*CBT-Net. We can see that *k*CBT-Net produces closer results to GT than SZ, especially on the VM (640) data set, which matches the quantitative results of PSNRs (RAEs).

Summary. Based on the above results, we conclude that Scalar2Vec outperforms the three baseline methods, Pix2Pix, Cy-



(a) Pix2Pix (b) CycleGAN (c) V2V (d) *k*CBT-Net (e) GT

Figure 9: Visualization results of filtered streamlines with different methods. Top to bottom: streamlines traced from translated vector fields with half-cylinder (VM (640)), hurricane (VM), supercurrent (RHO), and tornado (VM) input scalar fields. Out of the 500 streamlines shown in Figure 8, 5, 8, 14, and 21 streamlines are shown, respectively, from top to bottom. Seeding points are shown as green spheres. For half-cylinder (VM (640)), no seeding points are shown due to the cropped view.

cleGAN, and V2V. This demonstrates that (1) Scalar2Vec is more effective in dealing with the more challenging task (i.e., scalar-to-vector translation); (2) *k*CBT-Net is a more powerful model in exploiting multi-scale information and handing multivariate patterns. A parameter study on architecture design, module connection, initial channels, and loss functions is given in the Appendix.

4.2 Limitations and Future Work

Our Scalar2Vec work has the following limitations. First, although we have shown that Scalar2Vec can learn Scalar2Vec translation from not only VM but also other scalar fields, the current network needs to be trained again given a different input scalar variable. The only exception we make is the ensemble half-cylinder data set, but we witness a considerable performance drop. We will further explore how to generalize this framework to avoid training the network from scratch. Second, we simply treat time-dependent vector fields as sample data, randomly draw a certain percentage of samples for network training, and use the remaining samples for inference. As such, the framework currently does not consider unsteady vector fields, and we do not provide video clips showing temporally coherent inference results as V2V [23]. However, we will investigate how well the method operates in time-dependent scenarios in the future. Third, Scalar2Vec needs 40% of volume samples for effective training and inference. We will study how to improve the solution so that fewer samples are required. Fourth, although we show that Scalar2Vec can preserve physical quantities such as vorticity better than other methods, Scalar2Vec is not a physics-informed deep learning solution. Thus, incorporating the underlying physics (e.g., divergence-free properties) into the translation task calls for new solutions. Addressing these limitations will make Scalar2Vec a valuable alternative for domain scientists to recover the vector field data from their scalar field counterparts.

5 Conclusions

We have presented Scalar2Vec, a new deep learning approach that translates scalar fields to velocity vector fields. The proposed framework includes two steps: variable selection and field translation. Variable selection chooses variables from multivariate or ensemble scalar fields that are suitable for the translation task. Inspired by I2I translation and, in particular, image colorization, we design a *k*CBT-Net to achieve field translation. We have shown that Scalar2Vec outperforms other deep learning solutions (Pix2Pix, CycleGAN, and V2V), using quantitative and qualitative results. With the success of recovering vector fields from their scalar field counterparts, Scalar2Vec provides a new option for scientists to determine the desired data storage and I/O strategy for their simulations.

Acknowledgments

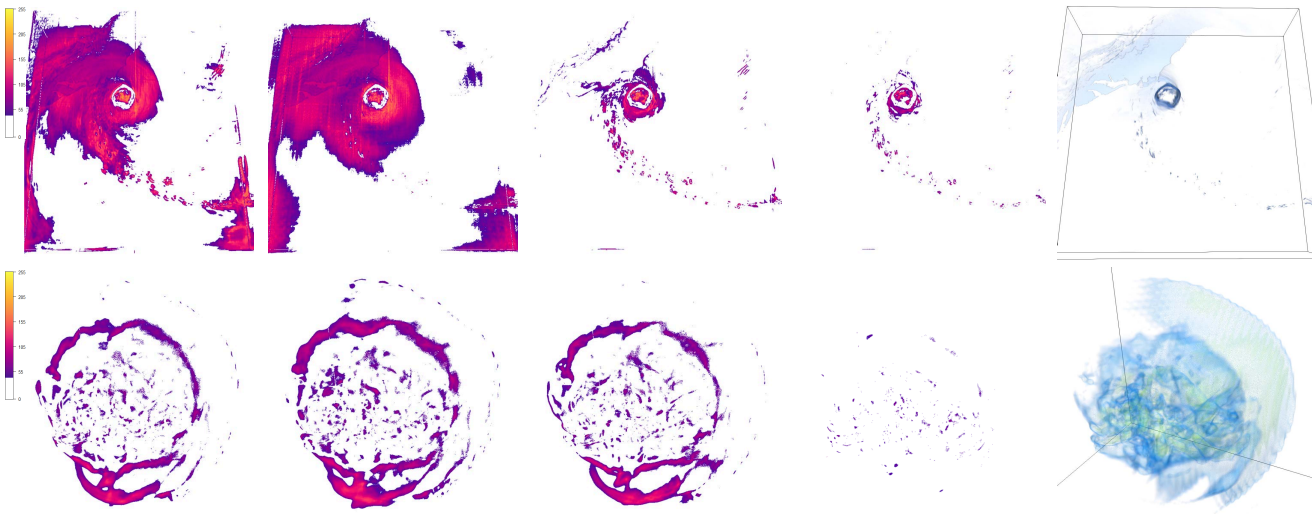
This research was supported in part by the U.S. National Science Foundation through grants IIS-1455886, CNS-1629914, DUE-1833129, IIS-1955395, IIS-2101696, and OAC-2104158. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] IEEE Visualization 2004 Contest. `http://vis.computer.org/vis2004contest/data.html`. Accessed: 2021-01-10.

[2] Y. An, H.-W. Shen, G. Shan, G. Li, and J. Liu. STSRNet: Deep joint space-time super-resolution for vector field visualization. *IEEE Computer Graphics and Applications*, 41(6):122–132, 2021.

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



(a) Pix2Pix (b) CycleGAN (c) V2V (d) *k*CBT-Net (e) GT

Figure 10: Vorticity visualization results with different methods. Top to bottom: hurricane (VM) and supernova (VM). (e) shows volume rendering of the vorticity scalar field derived from the GT velocity vector field. (a) to (d) show the pixel-wise difference images of vorticity visualizations from synthesized and GT data.

[4] J. M. Blondin and A. Mezzacappa. Pulsar spins from an instability in the accretion shock of supernovae. *Nature*, 445(7123):58–60, 2007.

[5] J. Chen, S. Banerjee, A. Grama, W. J. Scheirer, and D. Z. Chen. Neuron segmentation using deep complete bipartite networks. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 21–29, 2017.

[6] J. Chen, L. Yang, Y. Zhang, M. Alber, and D. Z. Chen. Combining fully convolutional and recurrent neural networks for 3D biomedical image segmentation. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 3036–3044, 2016.

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

[8] Z. Cheng, Q. Yang, and B. Sheng. Deep colorization. In *Proceedings of IEEE International Conference on Computer Vision*, pp. 415–423, 2015.

[9] Y. Ci, X. Ma, Z. Wang, H. Li, and Z. Luo. User-guided deep anime line art colorization with conditional adversarial networks. In *Proceedings of ACM International Conference on Multimedia*, pp. 1536–1544, 2018.

[10] R. A. Crawfis and N. Max. Texture splats for 3D scalar and vector field visualization. In *Proceedings of IEEE Visualization Conference*, pp. 261–267, 1993.

[11] A. Gonzalez-Garcia, J. v. d. Weijer, and Y. Bengio. Image-to-image translation for cross-domain disentanglement. In *Proceedings of Advances in Neural Information Processing Systems*, pp. 1294–1305, 2018.

[12] P. Gu, J. Han, D. Z. Chen, and C. Wang. Reconstructing unsteady flow data from representative streamlines via diffusion and deep learning based denoising. *IEEE Computer Graphics and Applications*, 41(6):111–121, 2021.

[13] P. Gu, H. Zheng, Y. Zhang, C. Wang, and D. Z. Chen. *k*CBAC-Net: Deeply supervised complete bipartite networks with asymmetric convolutions for medical image segmentation. In *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 337–347, 2021.

[14] 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*, pp. 71–80, 2020.

[15] 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.

[16] 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.

[17] 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. Accepted.

[18] 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.

[19] J. Han and C. Wang. SurfNet: Learning surface representations via graph convolutional network. *Computer Graphics Forum*, 2022. Conditionally Accepted.

[20] J. Han and C. Wang. TSR-VFD: Generating temporal super-resolution for unsteady vector field data. *Computers & Graphics*, 2022. Accepted.

[21] J. Han and C. Wang. VCNet: A generative model for volume completion. *Visual Informatics*, 2022. Conditionally Accepted.

[22] J. Han, H. Zheng, D. Z. Chen, and C. Wang. STNet: An end-to-end generative framework for synthesizing spatiotemporal super-resolution volumes. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):270–280, 2022.

[23] J. Han, H. Zheng, Y. Xing, D. Z. Chen, and C. Wang. V2V: A deep learning approach to variable-to-variable selection and translation for multivariate time-varying data. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1290–1300, 2021.

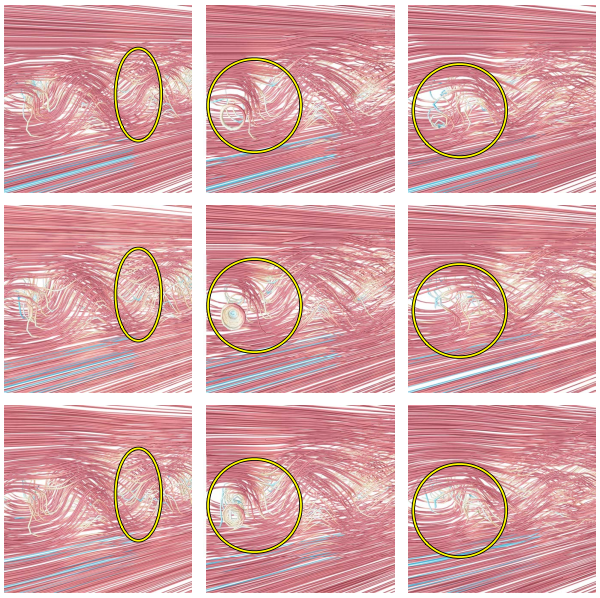
[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[25] M. He, D. Chen, J. Liao, P. V. Sander, and L. Yuan. Deep exemplar-based colorization. *ACM Transactions on Graphics*, 37(4):1–16, 2018.

[26] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. 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.

[27] 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.

[28] F. Hong, J. Zhang, and X. Yuan. Access pattern learning with long short-term memory for parallel particle tracing. In *Proceedings of*



(a) VM (320) (b) VM (640) (c) VM (6,400)

Figure 11: Visualization results of the half-cylinder data set with different methods using different VM ensembles. Top to bottom: streamlines traced from the SZ compressed then decompressed vector fields, streamlines traced from *k*CBT-Net translated vector fields, streamlines traced from GT vector fields.

IEEE Pacific Visualization Symposium, pp. 76–85, 2018.

[29] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of European Conference on Computer Vision*, pp. 172–189, 2018.

[30] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics*, 35(4):1–11, 2016.

[31] 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*, pp. 1125–1134, 2017.

[32] J. Jakob, M. Gross, and T. Günther. A fluid flow data set for machine learning and its application to neural flow map interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1279–1289, 2021.

[33] 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.

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

[35] 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 IEEE International Conference on Big Data*, pp. 438–447, 2018.

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

[37] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015.

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

[39] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[40] 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.

[41] 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*, pp. 8024–8035, 2019.

[42] S. Popinet, M. Smith, and C. Stevens. Experimental and numerical study of the turbulence characteristics of airflow around a research vessel. *Journal of Atmospheric and Oceanic Technology*, 21(10):1575–1589, 2004.

[43] I. B. Rojo and T. Günther. Vector field topology of time-dependent flows in a steady reference frame. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):280–290, 2019.

[44] 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*, pp. 234–241, 2015.

[45] M. Saito and Y. Matsui. Illustration2Vec: A semantic vector representation of illustrations. In *Proceedings of SIGGRAPH Asia 2015 Technical Briefs*, pp. 1–4, 2015.

[46] 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.

[47] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of International Conference on Learning Representations*, 2015.

[48] J.-W. Su, H.-K. Chu, and J.-B. Huang. Instance-aware image colorization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7968–7977, 2020.

[49] H. Tang, D. Xu, N. Sebe, Y. Wang, J. J. Corso, and Y. Yan. Multi-channel attention selection GAN with cascaded semantic guidance for cross-view image translation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2417–2426, 2019.

[50] Y.-L. Wang, A. Glatz, G. J. Kimmel, I. S. Aranson, L. R. Thoutam, Z.-L. Xiao, G. R. Berdiyorov, F. M. Peeters, G. W. Crabtree, and W.-K. Kwok. Parallel magnetic field suppresses dissipation in superconducting nanostrips. *Proceedings of the National Academy of Sciences of the United States of America*, 114(48):E10274–E10280, 2017.

[51] 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*, 27(6):3064–3078, 2021.

[52] W. Xian, P. Sangkloy, V. Agrawal, A. Raj, J. Lu, C. Fang, F. Yu, and J. Hays. TextureGAN: Controlling deep image synthesis with texture patches. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8456–8465, 2018.

[53] 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.

[54] Z. Xu, T. Wang, F. Fang, Y. Sheng, and G. Zhang. Stylization-based architecture for fast deep exemplar colorization. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9363–9372, 2020.

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

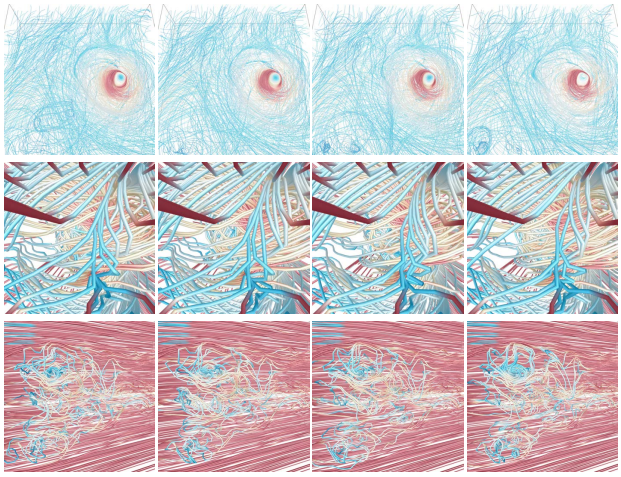
[56] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics*, 36(4):119, 2017.

[57] 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.

[58] 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*, pp. 2223–2232, 2017.

APPENDIX

Besides the main results presented in the paper, we conduct a parameter study to investigate the performance of Scalar2Vec, including architecture design, module connection, initial channels, and loss functions.



(a) CBT-Net (b) 2CBT-Net (c) 3CBT-Net (d) GT

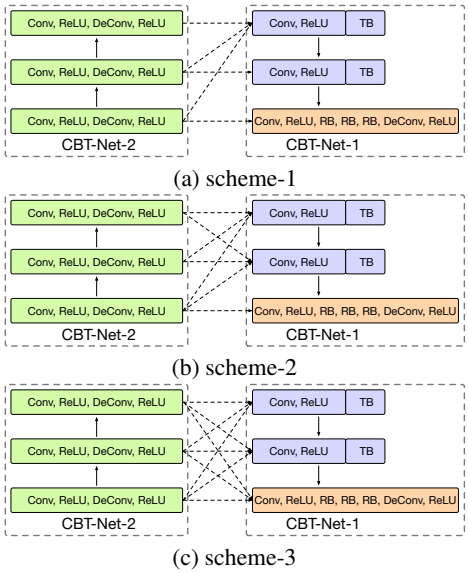
Figure 1: Streamline visualization results with different k for kCBT-Net. Top to bottom: streamlines traced from translated vector fields with hurricane (QVAPOR), supernova (VM), and Tangaroa (VM) input scalar fields.

Table 1: Average PSNR (dB) and RAE values, and model size (GB) using different architecture designs.

data set	architecture	PSNR	RAE	model size
hurricane (QVAPOR)	CBT-Net	40.13	0.406	0.401
	2CBT-Net	41.83	0.369	1.136
	3CBT-Net	**41.85**	**0.369**	1.871
supernova (VM)	CBT-Net	50.71	0.076	0.401
	2CBT-Net	51.13	0.071	1.136
	3CBT-Net	**51.14**	**0.071**	1.871
Tangaroa (VM)	CBT-Net	55.45	0.071	0.401
	2CBT-Net	56.44	0.065	1.136
	3CBT-Net	**56.53**	**0.065**	1.871

Architecture design. To study the impact of the number of modules for Scalar2Vec translation, we conduct experiments with the hurricane (QVAPOR), supernova (VM), and Tangaroa (VM) input scalar fields by setting different values (1 to 3) for k. We denote these different architectures as kCBT-Net when $k > 1$. In Figure 1, we show the streamlines traced from translated vector fields produced by different settings (i.e., kCBT-Net). We can see that the visual results do not change significantly with different architecture designs. As shown in Table 1, although 3CBT-Net achieves the best average PSNR and RAE values, its advantage over 2CBT-Net is slim. Considering the model size (which is data set independent), we select 2CBT-Net as a tradeoff. Referring to Table 3 in the paper, we can see that Scalar2Vec (using 2CBT-Net) outperforms V2V in terms of both PSNR and RAE. For hurricane (QVAPOR), the differences of PSNR and RAE are 2.55 and 0.055. For supernova (VM), the differences are 1.95 and 0.012. For Tangaroa (VM), the differences are 1.68 and 0.006. All these improvements confirm the effectiveness of kCBT-Net on exploring multi-scale information.

Module connection. A natural strategy to sequentially connect the sub-modules is to copy the result from CBT-Net-t to the commensurate layer in CBT-Net-$(t-1)$. There are different ways to achieve this: (1) kCBT-Net: as shown within the red dashed line in Figure 2



(a) scheme-1

(b) scheme-2

(c) scheme-3

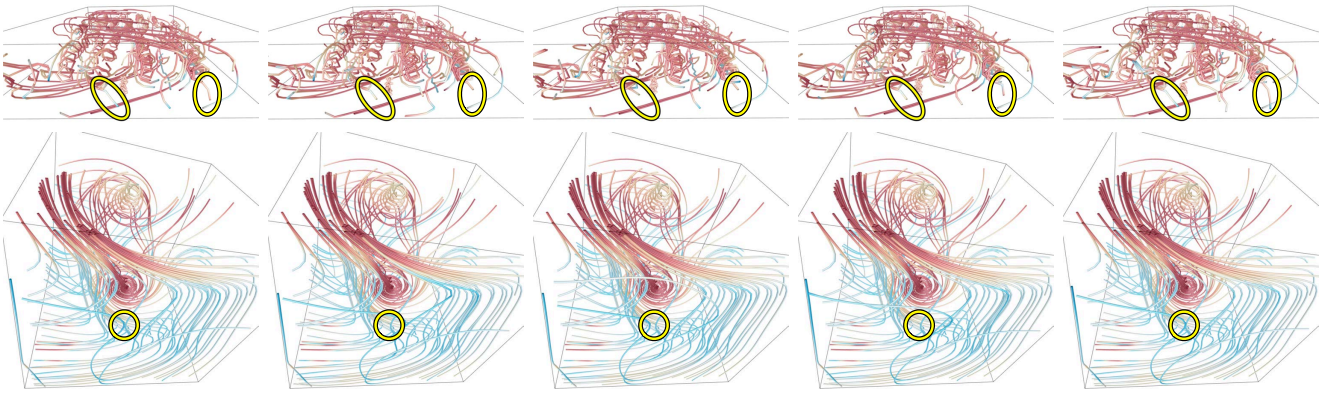
Figure 2: Three different module connection schemes, which are variants of the baseline connection shown within the red dashed line in Figure 2 of the paper.

Table 2: Average PSNR (dB) and RAE values, and model size (GB) using different module connection schemes.

data set	scheme	PSNR	RAE	model size
supercurrent (VM)	kCBT-Net	**62.92**	**0.059**	1.136
	scheme-1	62.70	0.059	1.141
	scheme-2	62.71	0.059	1.151
	scheme-3	62.69	0.059	1.163
tornado (VM)	kCBT-Net	**65.53**	**0.055**	1.136
	scheme-1	64.83	0.057	1.141
	scheme-2	64.99	0.057	1.151
	scheme-3	64.75	0.057	1.163

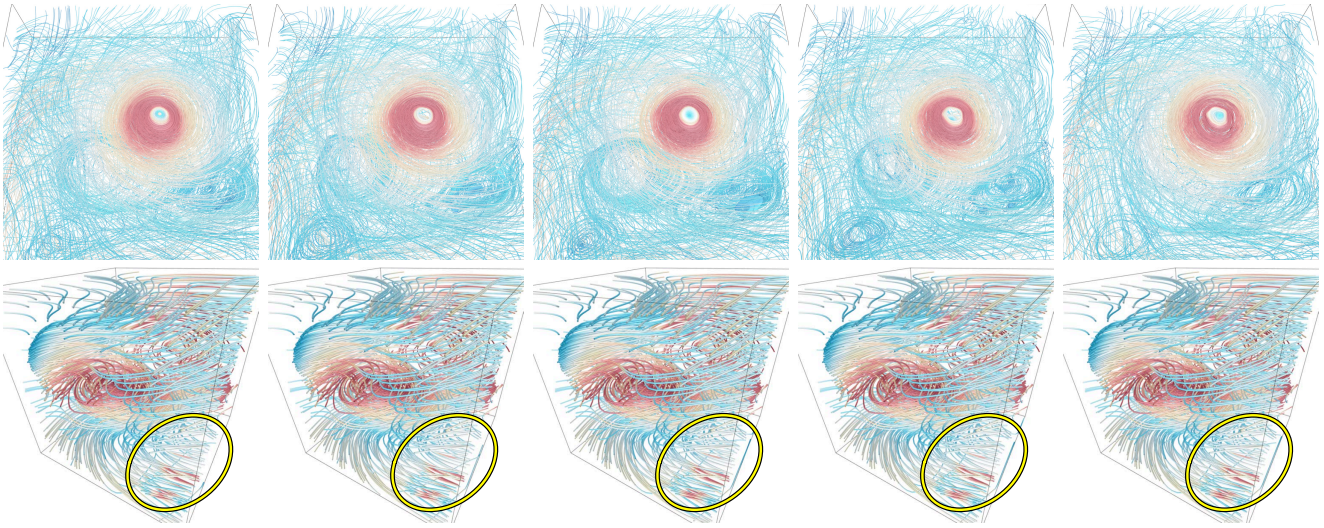
of the paper. we copy every piece of information from CBT-Net-t to CBT-Net-$(t-1)$ in the commensurate layers; (2) besides copy every piece of information from CBT-Net-t to CBT-Net-$(t-1)$ in the commensurate layers, we gradually copy all piece of information from CBT-Net-t to CBT-Net-$(t-1)$ until forming a fully-connected CBC structure, as illustrated in Figure 2 (a), (b), and (c), respectively. We denote these module connection schemes as scheme-1, scheme-2, and scheme-3, respectively. In Figure 3, we show the streamlines traced from translated vector fields generated using different module connection schemes. For a clear comparison, we show filtered streamline visualization results where we select a subset from the 500 streamlines that passes through high entropy regions. The highlights for both supercurrent (VM) and tornado (VM) results show that kCBT-Net leads to streamlines that are more similar to the GT ones. As shown in Table 2, kCBT-Net slightly outperforms the three schemes in terms of both PSNR and RAE. From the baseline kCBT-Net connection, the number of connections increases from scheme-1 to scheme-3, which leads to the increased model size. Therefore, we choose the baseline kCBT-Net connection instead of other schemes for Scalar2Vec.

Initial channels. For the number of initial channels c (refer to Table 1 in the paper), we experiment with different settings of 16, 32, 64, and 128 on the hurricane (QVAPOR) and tornado (VM) data sets. Figure 4 shows that the impact of using different c is not significant. Still, we can see from ellipse highlights for the tornado streamlines that using 64 initial channels achieves the most similar results compared with the GT streamlines. Table 3 shows that the



(a) kCBT-Net (b) scheme-1 (c) scheme-2 (d) scheme-3 (e) GT

Figure 3: Visualization results of filtered streamlines with different module connection schemes. Top and bottom: streamlines traced from translated vector fields with supercurrent (VM) and tornado (VM) input scalar fields. Out of the 500 streamlines, 150 and 100 streamlines are shown for these two data sets, respectively.



(a) 16 (b) 32 (c) 64 (d) 128 (e) GT

Figure 4: Streamline visualization results using different numbers of initial channels. Top and bottom: streamlines traced from translated vector fields with hurricane (QVAPOR) and tornado (VM) input scalar fields.

Table 3: Average PSNR (dB) and RAE values, and model size (GB) using different numbers of initial channels c.

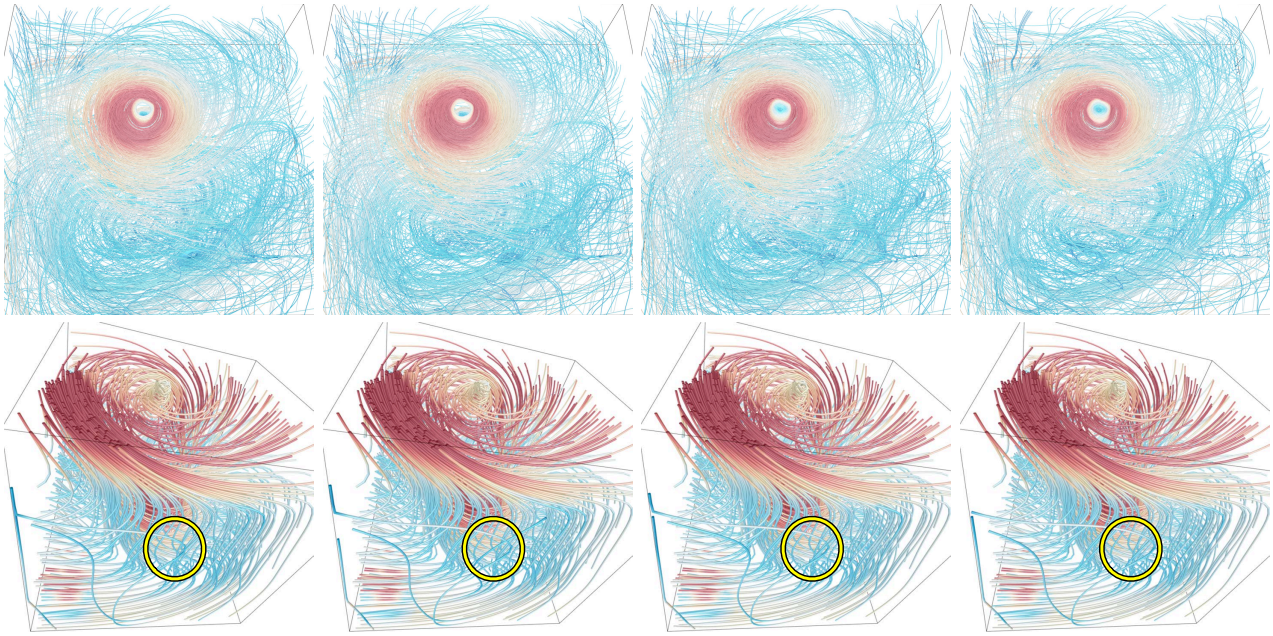
data set	c	PSNR	RAE	model size
hurricane (QVAPOR)	16	40.43	0.397	0.071
	32	41.40	0.377	0.284
	64	**41.83**	**0.369**	1.136
	128	41.56	0.376	4.541
tornado (VM)	16	55.63	0.085	0.071
	32	61.86	0.066	0.284
	64	65.53	0.055	1.136
	128	**65.55**	**0.055**	4.541

Table 4: Average PSNR (dB) and RAE values using different loss functions.

data set	loss functions	PSNR	RAE
hurricane (VM)	$\mathscr{L}_1 = \mathscr{L}_m$	45.83	0.287
	$\mathscr{L}_2 = \mathscr{L}_m + 10\mathscr{L}_j$	45.96	0.285
	$\mathscr{L}_3 = \mathscr{L}_m + 10\mathscr{L}_j + 0.005\mathscr{L}_a$	**46.27**	**0.281**
tornado (VM)	$\mathscr{L}_1 = \mathscr{L}_m$	64.15	0.057
	$\mathscr{L}_2 = \mathscr{L}_m + 10\mathscr{L}_j$	65.35	0.056
	$\mathscr{L}_3 = \mathscr{L}_m + 10\mathscr{L}_j + 0.005\mathscr{L}_a$	**65.53**	**0.055**

model size increases by four folds as c doubles. Considering both PSNR and RAE, we decide to use 64 initial channels as it turns out to be the overall best choice.

Loss functions. To explore the impact of different loss functions (i.e., the addition of angle loss ($\mathscr{L}_a$) and Jacobian loss ($\mathscr{L}_j$)) for Scalar2Vec translation, we conduct experiments with hurricane (VM) and tornado (VM) input scalar fields. Specifically, we start with magnitude loss $\mathscr{L}_m$ as $\mathscr{L}_1$, then add $\mathscr{L}_j$ to get $\mathscr{L}_2$, and finally add $\mathscr{L}_a$ to form the final loss ($\mathscr{L}_3$). We let $\mathscr{L}_m$ dominating the loss when determining the weights, and all the weights are empirically decided based on experiments. In Figure 5, we compare the streamlines visualization results with different loss functions. For the hurricane data set using the VM input scalar field, $\mathscr{L}_3$ (i.e., the combination of magnitude, angle, and Jacobian losses) produces the best results around the hurricane's eye (i.e., the innermost blue cluster of streamlines at the eye). The same observation can be obtained for the tornado data set using the VM input scalar field,



(a) $\mathscr{L}_1$ (b) $\mathscr{L}_2$ (c) $\mathscr{L}_3$ (d) GT

Figure 5: Streamline visualization results with different loss functions. Top to bottom: streamlines traced from translated vector fields with hurricane (VM) and tornado (VM) input scalar fields.

referring to the ellipse highlights. The improvements in Table 4 on both hurricane (VM) and tornado (VM) data sets indicate the effectiveness of the addition of angle and Jacobian losses.