Computers & Graphics (2023)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

STSR-INR: Spatiotemporal Super-Resolution for Multivariate Time-Varying Volumetric Data via Implicit Neural Representation

Kaiyuan Tang[a], Chaoli Wang[a]

[a] *Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556, United States*

ARTICLE INFO

Article history:
Received November 26, 2023

Keywords: Spatiotemporal super-resolution, implicit neural representation, multivariate time-varying data

ABSTRACT

Implicit neural representation (INR) has surfaced as a promising direction for solving different scientific visualization tasks due to its continuous representation and flexible input and output settings. We present STSR-INR, an INR solution for generating simultaneous spatiotemporal super-resolution for multivariate time-varying volumetric data. Inheriting the benefits of the INR-based approach, STSR-INR supports unsupervised learning and permits data upscaling with arbitrary spatial and temporal scale factors. Unlike existing GAN- or INR-based super-resolution methods, STSR-INR focuses on tackling variables or ensembles and enabling joint training across datasets of various spatiotemporal resolutions. We achieve this capability via a variable embedding scheme that learns latent vectors for different variables. In conjunction with a modulated structure in the network design, we employ a variational auto-decoder to optimize the learnable latent vectors to enable latent-space interpolation. To combat the slow training of INR, we leverage a multi-head strategy to improve training and inference speed with significant speedup. We demonstrate the effectiveness of STSR-INR with multiple scalar field datasets and compare it with conventional tricubic+linear interpolation and state-of-the-art deep-learning-based solutions (STNet and CoordNet).

1. Introduction

In many applications, domain scientists run large-scale simulations to generate spatiotemporal multivariate volumetric data for analyzing the corresponding physical or chemical processes. These simulations often come with various conditions, settings, or configurations, leading to multiple runs. The resulting multivariate or ensemble data are different but usually share a similar structural appearance. Analyzing and visualizing such high-dimensional spatiotemporal data requires enormous disk and memory storage for post hoc analysis, presenting a significant challenge to domain experts and visualization researchers.

One way to tame the high storage cost is to save only downsampled low-resolution data and then apply spatiotemporal super-resolution (STSR) techniques to recover their high-resolution counterparts. For instance, given a downsampled volume sequence (e.g., 50 timesteps with 128^3 spatial resolution), the STSR task aims to upsample the sequence to a high-resolution one (e.g., 150 timesteps with 512^3 spatial resolution). Over the past few years, we have witnessed a surge of deep-learning-based solutions for accomplishing many scientific visualization tasks, including super-resolution generation [1]. For the end-to-end STSR generation, STNet [2] and STSRNet [3] are state-of-the-art examples that upscale volumetric scalar and vector data, respectively. Nevertheless, both works suffer significant limitations.

First, these solutions are based on convolutional neural networks (CNNs) and generative adversarial networks (GANs). Due to their *discrete, resolution-dependent* network designs, CNN and GAN-based STSR solutions demand ground-truth (GT) high-resolution data during training in a supervised manner. They cannot interpolate arbitrarily-resolved spatial or temporal resolution.

Second, neither STSR method provides sound guidance for training *multivariate or ensemble* datasets. They tackle each variable or ensemble sequence as an independent training process, making similar structure learning redundant. One straightforward way to achieve multivariate STSR is to expand the network's output, i.e., inferring multiple variables simultaneously. This calls for an increased network capacity, which may not always be desirable. Moreover, the variation of variable or ensemble distributions could negatively impact each other during training, leading to performance degradation.

Third, moving from different variables to *different datasets*, both STNet and STSRNet do not permit joint training of different datasets of various spatiotemporal resolutions. For flexibility and efficiency, it is ideal that the same network trains multiple datasets simultaneously without sacrificing inference quality. However, such a joint training scheme has not been thoroughly studied in scientific visualization for the STSR task.

To respond, we design STSR-INR, **s**patio**t**emporal **s**uper-**r**esolution via **i**mplicit **n**eural **r**epresentation. Unlike CNN or GAN, INR ingests coordinates and predicts quantities of interest via a neural network, commonly in the form of *multilayer perceptrons* (MLPs) or *fully-connected network* (FCN). With INR, the memory required to parameterize the signal depends on its *complexity* rather than *resolution*. We leverage such an FCN to learn a *continuous* representation from *discrete* data samples. Doing so brings two benefits. First, it can achieve *unsupervised learning*, which does not need seeing low- and high-resolution volume pairs for training. Second, it supports upsampling the input low-resolution volume sequence to an *arbitrary spatial or temporal scale* without modifying network structure or retraining.

To help the network learn multivariate sequences, we design a *variable embedding* scheme along with a *modulated structure* to optimize each variable independently while utilizing their shared structural appearance for training. Variable embedding models each variable or ensemble sequence as a *learnable latent vector*, enabling the network to capture more detailed variable variations. To better utilize the latent vector, we devise a modulated structure consisting of a *modulator network* and a *synthesis network*. The modulator network will provide the latent vector with more control over the feature map in the synthesis network and, thus, could improve the quality of synthesized spatiotemporal volumes. This embedding structure is highly flexible and can support the joint training of different datasets of various spatiotemporal resolutions within the same network. Furthermore, our variable embedding is learned with a *variational auto-decoder*, which optimizes the latent vector, allowing us to conduct latent-space interpolation. Finally, INR-based solutions are notoriously slow in training as an entire feedforward pass through the network must be computed for each sample. We utilize a *multi-head strategy* to boost the training and inference speed of STSR-INR significantly.

We experiment with STSR-INR on several multivariate or ensemble scalar field datasets and compare it against tricubic+linear interpolation, GAN-based STNet [2], and INR-based CoordNet [4]. The results demonstrate that STSR-INR achieves competitive quality on most datasets using data-, image-, and feature-level metrics. The contribution of this paper is as follows. First, we present the design of STSR-INR, a novel INR-based solution to achieve STSR for multivariate or ensemble spatiotemporal volume data. Second, we experiment with the multi-head strategy to effectively tackle the issue of slow training with INR. Third, we investigate the utility of our embedding structure via joint training, latent-space interpolation, and network analysis. Fourth, we show the advantages of STSR-INR over the state-of-the-art STSR solutions based on GAN and INR. Finally, we investigate two key network settings for STSR-INR and study their impacts on performance.

2. Related work

This section discusses related works of deep learning for scientific visualization, super-resolution generation, and INR techniques.

2.1. Deep Learning for Scientific Visualization

There is an exciting trend of leveraging deep-learning-based methods for solving scientific visualization tasks, including data generation, visualization generation, prediction, object detection and segmentation, and feature learning and extraction [1]. Among them, the task most relevant to this work is *data generation*, which aims to infer or reconstruct new versions of data from existing versions or their reduced visual representations. The most popular form of data generation is *super-resolution generation*, which uses downsampled low-resolution data to produce high-resolution versions [5]. For instance, Han and Wang designed SSR-TVD [6], which applies a GAN to upscale the low-resolution 3D volumetric sequences into high-resolution ones. Another form of data generation is *data reconstruction*, which infers the original data from their visual representations. For example, Gu et al. [7] considered the problem of reconstructing unsteady flow data from their reduced visual forms: a set of representative streamlines. Their VFR-UFD solution can recover high-quality vector data from these compact streamlines via a diffusion step followed by deep-learning-based denoising. The third form of data generation is *data translation*. i.e., ingesting one variable or ensemble sequence to infer another sequence, commonly called variable-to-variable (V2V) translation [8]. For instance, Scalar2Vec [9] translates one scalar field to its corresponding velocity vector field using the *k*-complete bipartite translation network.

In this work, we focus on spatiotemporal super-resolution generation. Given low spatial and temporal resolution volume sequences, we aim to upscale them to high spatial and temporal resolution ones in an end-to-end fashion, similar to STNet [2]. Unlike vector field STSR model STSRNet [3], STSR-INR is designed for scalar field multivariate time-varying data and does not upscale vector field data or include motion estimation for flow field reconstruction. One of the concurrent works is FFEINR [10], which employs INR to achieve STSR for flow field data with fast training and inference speed. In contrast to FFEINR, which utilizes an encoder to extract the downscaled data features, our STSR-INR embeds features through a series of learnable latent vectors and thus could maintain a relatively lightweight architecture.

2.2. Super-Resolution Generation

Super-resolution techniques transform low-resolution data into high-resolution versions, including spatial super-resolution (SSR), temporal super-resolution (TSR), and STSR. Examples of deep-learning-based SSR works are SRCNN [11], SRFBN [12], and SwinIR [13], which solve the inference of high-resolution details in the spatial domain. TSR takes subsampled time sequences to interpolate intermediate timesteps with the same spatial resolution. Example works include phase-based interpolation [14], SepConv [15], and SloMo [16]. SSR and TSR only focus on the spatial or temporal domain, but not both. STSR addresses both spatial and temporal super-resolution simultaneously. Compared with conventional interpolation methods, deep-learning-based methods can reconstruct more accurate results because of their ability to fit complex global patterns of the target data.

Our work falls into the STSR category. Previous methods, like STNet [2], can only upsample the input data with a *fixed* spatial or temporal scale factor. On the contrary, our STSR-INR can upscale the input low-resolution data to an *arbitrary* scale, thanks to the *continuous* neural representation of spatial and temporal domains. Moreover, STSR-INR accomplishes spatial and temporal upscaling in an *unsupervised* manner. This means that, unlike STNet, STSR-INR does not keep low- and high-resolution volume pairs or the complete subsequence of early timesteps for training optimization.

2.3. INR-Based Techniques

Recent works have investigated utilizing MLPs or FCNs to learn the continuous INR from discrete data samples. The most notable works are neural radiance field (NeRF) and sinusoidal representation network (SIREN). Mildenhall et al. [17] introduced NeRF, which applies an FCN with position encoding to learn the continuous volumetric scene and synthesize novel views. Sitzmann et al. [18] proposed SIREN that leverages the periodic activation function to help the MLPs learn the complex data signals more accurately. In scientific visualization, INR-based examples include neurcomp for neural compression of volume data [19], fV-SRN, a fast version of a scene representation network for volume rendering [20], neural flow map for particle trajectory prediction [21], and instant neural representation for interactive volume rendering [22].

Researchers have extracted feature information and injected it into the INR model's input to improve the performance and generalization ability. A direction is utilizing an encoder to extract latent features from subsampled data, often called the auto-encoder architecture. Example works in this direction include VideoINR [23] and ArSSR [24]. However, volumetric data are massive 4D space-time data, often demanding excessive GPU memory consumption when applying an encoder for feature extraction. Instead of using the *auto-encoder* architecture, we leverage the *auto-decoder* architecture, which derives the feature information by assigning each type of signal (e.g., variable or ensemble) a learnable latent vector and optimizing the latent vector together with deep network parameters in the training process. Works such as DeepSDF [25] and DyNeRF [26] fall into this category.

The work most closely related to our work is CoordNet [4], which leverages INR to achieve data generation (i.e., SSR and TSR) and visualization generation (i.e., view synthesis and ambient occlusion prediction) tasks. Our STSR-INR work also targets super-resolution generation via INR. However, it tackles SSR and TSR simultaneously. We make significant changes to the baseline CoordNet framework to efficiently and effectively handle super-resolution generation for multiple variables or ensembles, which has never been explored. Furthermore, instead of training each model to learn the representation of individual datasets, our work can train the same model to learn across multiple datasets with various spatiotemporal resolutions. A recent concurrent work that also adopts CoordNet is HyperINR [27], which employs hypernetwork to produce the weights of an INR. However, HyperINR mainly focuses on the TSR of one scalar dataset, while our STSR-INR processes STSR on a single multivariate dataset or across multiple datasets.

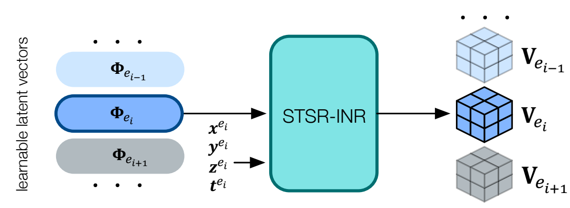


Fig. 1: Overview of STSR-INR. The network predicts the corresponding voxel value by inputting the variable-specific latent vector and space-time coordinates.

3. STSR-INR

3.1. Overview

Let $\mathbf{D} = \{\mathbf{D}_{e_1}, \mathbf{D}_{e_2}, \dots, \mathbf{D}_{e_n}\}$ be a set of n multivariate volume sequences, where $\mathbf{D}_{e_i}$ is the volume sequence for variable or ensemble e_i and $\mathbf{e} = \{e_1, e_2, \dots, e_n\}$. $\mathbf{D}_{e_i} = \{\mathbf{C}_{e_i}, \mathbf{V}_{e_i}\}$ contains a set of input space-time coordinates $\mathbf{C}_{e_i} = \{(x_1^{e_i}, y_1^{e_i}, z_1^{e_i}, t_1^{e_i}), (x_2^{e_i}, y_2^{e_i}, z_2^{e_i}, t_2^{e_i}), \dots\}$ and their corresponding values $\mathbf{V}_{e_i} = \{v_1^{e_i}, v_2^{e_i}, \dots\}$. As sketched in Figure 1, to achieve simultaneous training over multiple variables, we design *variable embedding* that assigns each variable sequence $\mathbf{D}_{e_i}$ a *learnable latent vector* Φ_{e_i}. During training, we aim to learn the mapping from $\mathbf{C}_{e_i}$ conditioned on Φ_{e_i} to $\mathbf{V}_{e_i}$ by updating both Φ_{e_i} and network parameters Θ. That is,

$$\mathcal{F}_\Theta : (\mathbf{C}; \Phi) \to \mathbf{V}, \tag{1}$$

where $\mathbf{C} = \{\mathbf{C}_{e_1}, \mathbf{C}_{e_2}, \dots, \mathbf{C}_{e_n}\}$, $\Phi = \{\Phi_{e_1}, \Phi_{e_2}, \dots, \Phi_{e_n}\}$, and $\mathbf{V} = \{\mathbf{V}_{e_1}, \mathbf{V}_{e_2}, \dots, \mathbf{V}_{e_n}\}$. Once the network is trained, given the optimized latent vector Φ_{e_i}, STSR-INR can predict $\mathbf{V}_{e_i}$ from unseen intermediate spatial and temporal coordinates. For the STSR task, given the spatial and temporal upscale factors u_s and u_t, it can reconstruct volume sequences with higher spatial and temporal resolutions by inference on a scaled spatiotemporal grid.

3.2. Network Architecture

SIREN and skip-connection. As illustrated in the left of Figure 2, our STSR-INR is a SIREN-based [18] network which

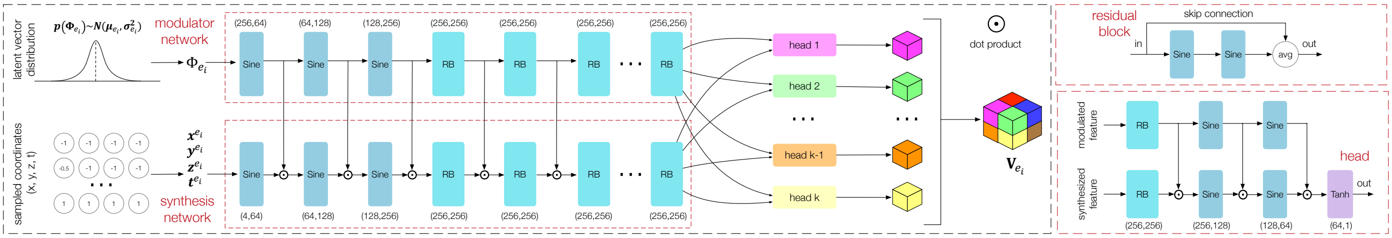


Fig. 2: Network structure of STSR-INR. Left: Overview of STSR-INR. The modulator network takes the sampled latent vector, and the synthesis network takes coordinates as input. The modulator then modulates the synthesis activations using dot products. Finally, each head in the multi-head structure reconstructs a subvolume of the same size in the whole volume. During training, we jointly optimize network and variable-specific latent vector distribution parameters. Top-right: Detailed structure of the residual block. Bottom-right: Detailed structure of the network's head part.

consists of fully-connected layers and the Sine activation function. Compared with other activation functions like ReLU or Tanh, employing Sine helps the network fit complex signals, especially high-frequency parts, more quickly and accurately. Moreover, if the input and output dimensions are consistent, we add skip-connection between every two consecutive SIREN layers to improve the network's capacity. These skip-connection blocks are referred to as *residual blocks*. Figure 2 top-right shows how the residual block is constructed. Following CoordNet [4], we also apply average operations on the residual block. For example, let the input of the residual block be $\mathbf{x}$, and $f(\mathbf{x})$ be the activation after two SIREN layers. The output of the residual block is $0.5(\mathbf{x}+f(\mathbf{x}))$. By multiplying 0.5 on the skip-connection result, the output range of one residual block stays in [-1,1] (which is the same as the input range) instead of [-2,2]. This treatment can stabilize network training.

Variable embedding. Training each variable with a separate neural network is not flexible or efficient in achieving STSR for a single multivariate dataset. Ideally, we want the network to support *joint training* even for different multivariate datasets with various spatiotemporal resolutions while utilizing their shared structural information to speed up network training. By supporting joint training of multivariate datasets end-to-end with one model, we can simplify the training pipeline and avoid storing duplicate models for different variables.

Inspired by temporal embedding for video synthesis [26], we devise *variable embedding* representing the reconstruction context of different variables under the joint training scenario. It embeds each variable sequence via a real-valued optimizable latent vector $\Phi_{e_i} \in \mathbb{R}^l$, where l denotes the latent vector's length. When we train on the STSR task, we jointly optimize variable-specific latent vectors Φ and network parameters Θ.

In Equation 1, even though all latent vectors Φ have the same length of l, each e_i can have its own input coordinates $\mathbf{C}_{e_i}$. Therefore, denoted by $\mathcal{F}_\Theta$, the network can train variables with different spatial or temporal resolutions jointly. One intuitive way to encode the reconstruction task's context information for each variable is to apply *one-hot vector*. Such a vector has a fixed length containing n bits of 0 and 1. For the one-hot vector of e_i, only the i-th bit is 1, and the rest are 0. Compared to one-hot vectors, our variable embedding offers the following benefits. First, one-hot vectors are a *disentangled* form of representation, but our learnable latent vectors are *distributed*. The length of our latent vectors does not increase as the number of variables increases. Second, one-hot vectors can only be *orthogonal* to each other, but our learnable latent vectors are updated during training to implicitly encode *differentiations* among variables. Once trained, variable embedding can describe the distribution difference between variables. Third, variable embedding has the potential to provide an *operable* latent space, where interpolating between these optimized latent vectors can infer novel results with an appropriate decoder.

Modulated structure. Unlike CoordNet, STSR-INR needs to embed variable information into different variable-specific latent vectors. To this end, designing a way to condition the network output with different latent vectors is necessary. An intuitive way to condition learnable latent vectors on the generative network is to concatenate the latent vectors with coordinates as input to the INR model. Works like DeepSDF [25] follow this path. However, as pointed out by Mehta et al. [28], the *concatenation* approach is less expressive compared with a *modulation* approach. Concatenating the latent vectors with input only changes the *phase* of the feature map, while modulation allows the latent vectors to control the *phase*, *frequency*, and *amplitude* of the feature maps. Therefore, we present a *modulated structure* that consists of a *modulator network* and a *synthesis network*. The modulator network ingests the variable embedding latent vector information and modulates the synthesis network via dot product the activations of each block in the two networks. Instead of using ReLU in the modulator network [28], we utilize Sine as the non-linear function and keep the same network structure for both networks. As a result, each layer's input and output ranges in the synthesis network remain [-1,1]. This adaptation leads to fast and stable network training.

Variational auto-decoder. During training, we jointly optimize network parameters Θ and conditioned learnable latent vectors Φ. After that, the INR model can take trained Φ to reconstruct variable sequences. In this case, Φ can be considered dimensionality reduction resulting from a representation learning process of high-dimensional multivariate data. We can leverage the optimized Φ to generate new latent vectors and infer results through the network or analyze the difference between variable sequences.

As such, we leverage *variational auto-decoder* (VAD) [29] that employs a strong regularization on variable embedding. VAD brings two benefits. First, the learned latent space could be more compact as it follows a standard normal distribution. Second, we can use sampled unseen latent vectors to infer smoother novel results due to the continuous and probabilistic nature of the VAD latent space. Similar to *variational auto-*

encoder (VAE) [30], for variable e_i, instead of using one non-random vector, we sample latent vector Φ_{e_i} through its optimizable posterior distribution $N(\mu_{e_i}, \sigma^2_{e_i})$. The distribution of Φ_{e_i} cannot be optimized directly, so we apply the reparameterization trick [30]: $\Phi_{e_i} = \mu_{e_i} + \sigma_{e_i} \bigodot \varepsilon$, where ε is sampled from $\varepsilon \sim N(0, \mathbf{I})$ and $\mathbf{I}$ is the identity matrix. Then we can optimize the distribution of Φ_{e_i} by optimizing μ_{e_i} and σ_{e_i}. However, our experiment shows that the optimizable σ_{e_i} could lead to unstable training. Therefore, we only make μ_{e_i} learnable and set σ_{e_i} close to a constant unit vector. Such an adaptation stabilizes the training of VAD. We remove the random component in Φ_{e_i} during inference and use μ_{e_i} to represent each variable. The key motivation for sampling latent vectors during training is to ensure the input latent space of the decoder is a continuous representation instead of a discrete one like *auto-decoder* (AD). By sampling latent vectors and optimizing their distributions in the latent space, interpolated latent vectors are less likely to fall out of the continuous latent space that the decoder can decode, leading to a more meaningful reconstruction.

Multi-head training. INR methods consisting of only the MLP structure usually suffer in the speeds of training and inference. This is because the model needs to iterate all the samples sequentially. One way to solve this problem is to replace part of the fully-connected layers in INR with several convolutional layers so the network output can be a whole volume/image instead of a single voxel/pixel. Works like NeRV [31] follow this route. However, according to the experimental results in [4], utilizing CNNs directly in the reconstruction task yields blurry and noisy prediction results. Inspired by the *multi-head structure* proposed by Aftab et al. [32], we leverage a multi-head strategy to significantly speed up the training without losing much of the reconstruction ability. Specifically, we partition the spatial volume into blocks of equal size and feed the network with local coordinates. These coordinates can be mapped to the voxel values of corresponding positions within each block. As sketched in the left of Figure 2, after applying multiple heads at the end of the network structure, each network's feedforward pass can output all values corresponding to the same local coordinate across all these spatial partitions. This strategy dramatically reduces the necessary floating point operations for reconstructing the whole signal. Furthermore, y considering the difference of volume blocks in each partition, each head of our network utilizes its independent parameters to process the shared features output by the modulator and synthesis networks. This treatment ensures that the network can fit the volume sequence efficiently. Figure 2 bottom-right shows how the head part is constructed. Compared to the standard one-head training, our network requires proportionally fewer feedforward passes as the number of heads increases.

3.3. Optimization

In the training process, we jointly optimize network parameters Θ and learnable latent vectors Φ. The objective function consists of two parts: *reconstruction* loss and *Kullback-Leibler divergence* (KLD) loss [33]. The total loss $\mathcal{L}$ is given by

$$\mathcal{L} = \mathcal{L}_{\text{REC}} + \lambda \mathcal{L}_{\text{KLD}}, \tag{2}$$

where $\mathcal{L}_{\text{REC}}$ and $\mathcal{L}_{\text{KLD}}$ are the reconstruction and KLD losses, and $\lambda \in [0, 1]$ controls the weight of the KLD term.

Reconstruction loss. Given the input coordinates and latent vectors, the network predicts the corresponding voxel values $\mathbf{V}_{\text{PRE}}$. Let the ground-truth voxel values be $\mathbf{V}_{\text{GT}}$ (in our case, low-resolution volumes), and the reconstruction loss is defined as

$$\mathcal{L}_{\text{REC}} = \|\mathbf{V}_{\text{PRE}} - \mathbf{V}_{\text{GT}}\|_2. \tag{3}$$

KLD loss. Like VAE [30], we add a KLD term to regularize the variable latent space and secure plausible interpolation results among various latent vectors. Let the prior distribution of the latent vector Φ_{e_i} be $q(\Phi_{e_i}|\mathbf{D}_{e_i})$. The KLD term is defined as

$$\mathcal{L}_{\text{KLD}} = \sum_{i=1}^{n} \text{KLD}\left(q(\Phi_{e_i}|\mathbf{D}_{e_i}) || p(\mathbf{D}_{e_i}|\Phi_{e_i})\right). \tag{4}$$

In the following experiment, we set our prior distribution $q(\Phi_{e_i}|\mathbf{D}_{e_i})$ as the Gaussian distribution where $p(\mathbf{D}_{e_i}|\Phi_{e_i})$ is the distribution of sampled latent vector. Therefore, the expanded form of Equation 4 then becomes

$$\mathcal{L}_{\text{KLD}} = \frac{1}{2} \sum_{i=1}^{n} (\sigma^2_{e_i} + \mu^2_{e_i} - 1 - \log \sigma^2_{e_i}). \tag{5}$$

For stable training, we do not optimize σ_{e_i} and only update μ_{e_i}. The value of μ_{e_i} is randomly initialized and the constant value of σ_{e_i} is set by making $\log \sigma^2_{e_i} = 10^{-3}$. As such, the effect of the KLD loss is to regulate the different latent vector distributions to be close in the latent space. This ensures that the interpolated latent vector among them is less likely to fall outside the meaningful latent space region the decoder can decode. However, the decoder may suffer in the optimization process when different distributions are too close. Therefore, we introduce λ in Equation 2 to control the strength of KLD regulation. Algorithm 1 outlines the STSR-INR training process.

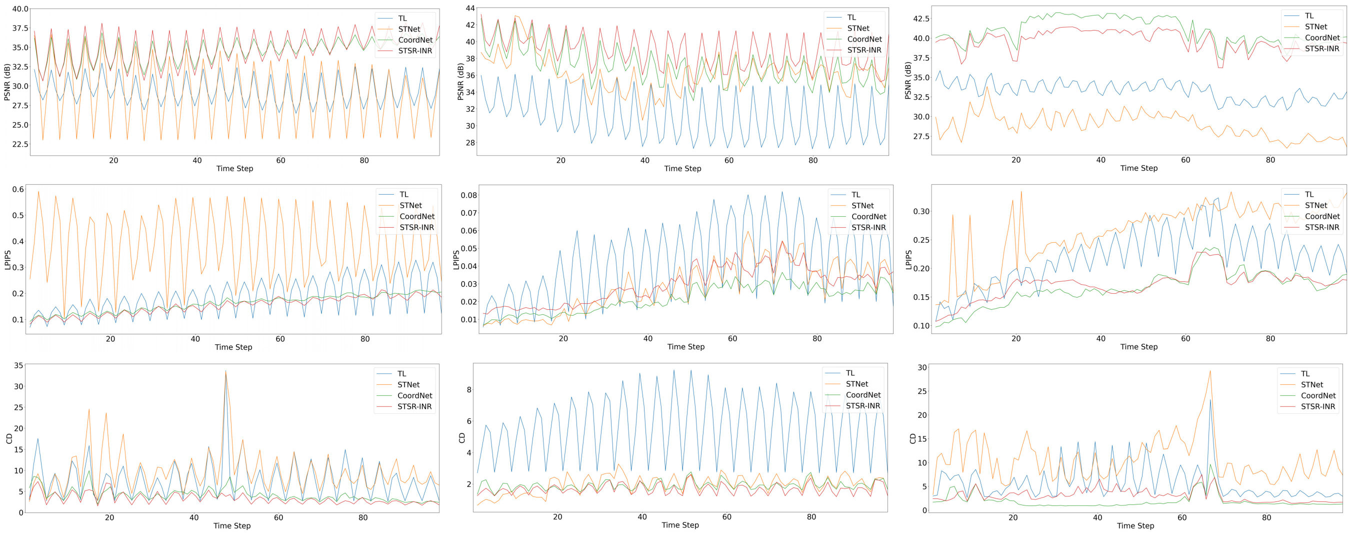
Algorithm 1 STSR-INR training algorithm

Input: Dataset $\mathbf{D} = \{(\mathbf{C}_{e_i}, \mathbf{V}_{e_i})\}_{i=1}^{n}$, target training epochs T, distribution variances of latent vectors $\sigma^2 = \{\sigma^2_{e_1}, \sigma^2_{e_2}, \ldots, \sigma^2_{e_n}\}$, KLD loss weight λ
  Randomly initialize network parameters Θ and distribution means of latent vectors $\mu = \{\mu_{e_1}, \mu_{e_2}, \ldots, \mu_{e_n}\}$
  for $j = 1 \ldots T$ **do**
    for all $(\mathbf{C}_{e_i}, \mathbf{V}_{e_i}) \in \mathbf{D}$ **do**
      $\Phi_{e_i} \leftarrow \mu_{e_i} + \sigma_{e_i} \bigodot \varepsilon$, where $\varepsilon \sim N(0, \mathbf{I})$
      Calculate $\mathbf{V}_{\text{PRE}} \leftarrow \mathcal{F}_{\Theta}(\mathbf{C}_{e_i}; \Phi_{e_i})$
      Compute $\mathcal{L}_{\text{REC}}$ following Equation (3)
      Update (Θ, μ) based on $\mathcal{L}_{\text{REC}}$
    end for
    Compute $\mathcal{L}_{\text{KLD}}$ following Equation (5)
    Update μ based on $\lambda \mathcal{L}_{\text{KLD}}$
  end for

4. Results and Discussion

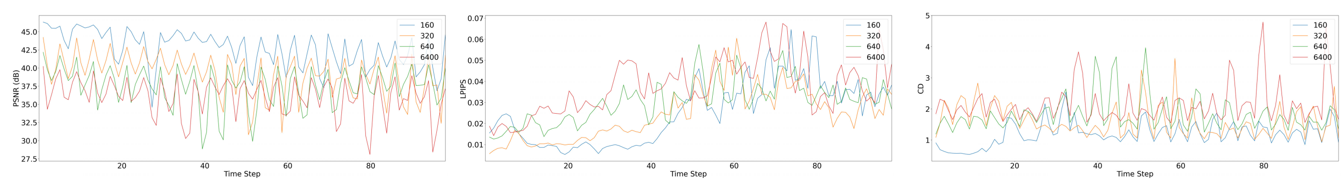
4.1. Datasets and Network Training

Datasets. Table 1 lists the datasets used in our experiments. The variable set of the combustion dataset [34] includes



(a) combustion (HR, MF, VTM, YOH) (b) half-cylinder (VLM: 160, 320, 640, 6400) (c) ionization (T, PD, H2, H+)

Fig. 3: Average PSNR (dB, top row), LPIPS (middle row), and CD (bottom row) values over the experimented variables of the three datasets.



(a) PSNR (b) LPIPS (c) CD

Fig. 4: PSNR (dB), LPIPS, and CD values for individual ensemble members of the half-cylinder (VLM) dataset using STSR-INR.

heat release (HR), mixture fraction (MF), vorticity magnitude (VTM), and OH mass fraction (YOH). The half-cylinder ensemble dataset [35] was produced from a fluid simulation under different Reynolds numbers (160, 320, 640, 6400). We use velocity magnitude (VLM). The ionization dataset [36] was produced from 3D radiation hydrodynamical calculations of the ionization front instabilities, and we use five variables: gas temperature (T), total particle density (PD), and mass abundances of H+, H2, and He. The Tangaroa dataset [37] has four variables: acceleration (ACC), divergence (DIV), VLM, and VTM. Finally, we also use three single-variable datasets: five-jet, tornado, and vortex, for additional experiments. Their variables are energy (E), VLM, and VTM.

Network training. The training and inference were performed on a single NVIDIA Tesla P100 graphics card with 16 GB of memory. The input spatial and temporal coordinates and target volume values were normalized to $[-1,1]$. The modulator network and synthesis network both have five residual blocks. We leverage 8-head in our network structure to achieve a trade-off between quality and speed (refers to Section 4.5). The network parameters were initialized following Sitzmann et al. [18]. Note that when applying the multi-head strategy to our STSR-INR model, we set the hyperparameter $\omega_0 = 5$ according to Yüce et al. [38]. This is to avoid utilizing aliased higher-frequency components for volume reconstruction with a low-sampling frequency. We set the batch size as 8000 total sampling points across all variables. We used Adam with a learning rate of 10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and L_2 weight decay of 10^{-6}. The weight of KLD loss λ was set to 10^{-3}. We trained STSR-INR for 600 epochs to converge.

Table 1: Variables and resolution of each dataset.

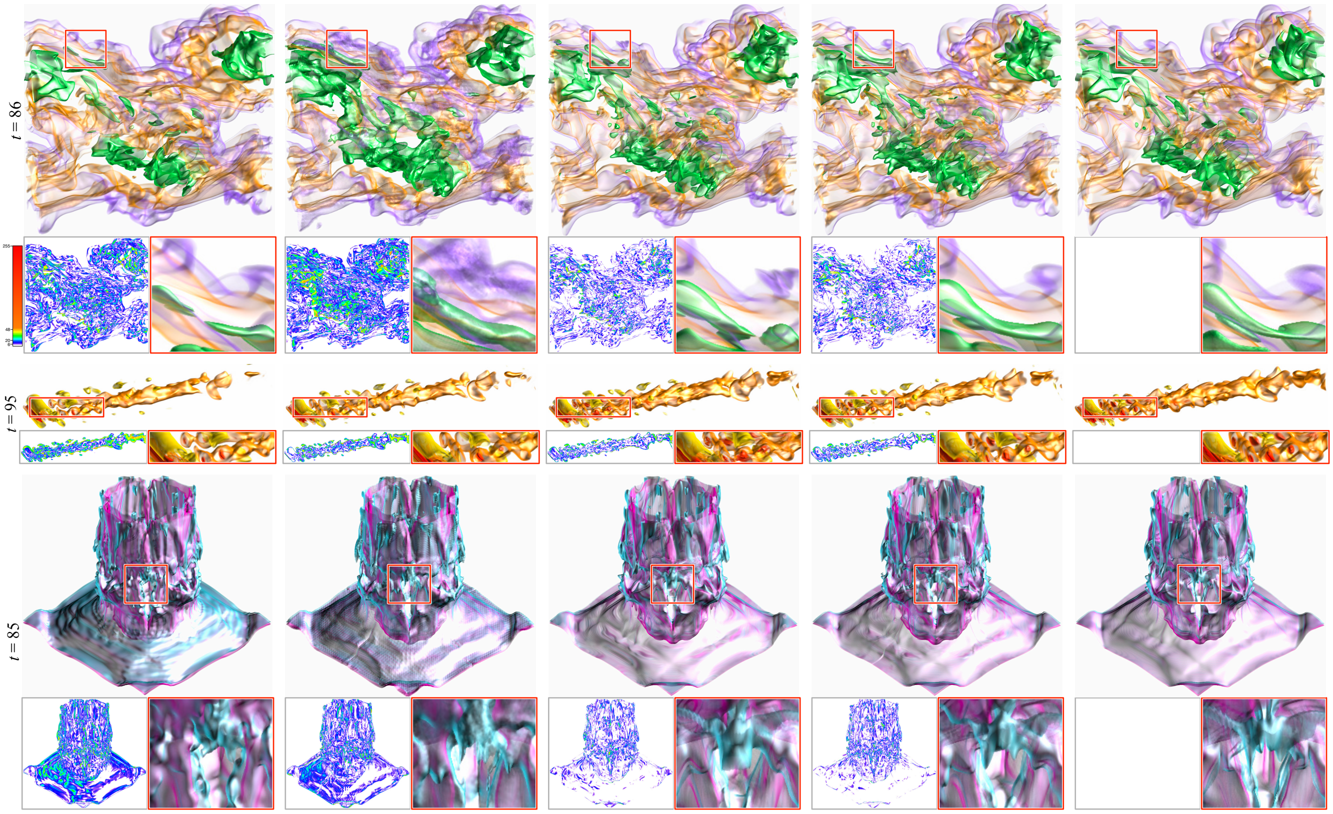
dataset	variables or ensembles	resolution ($x \times y \times z \times t$)
combustion [34]	HR, MF, VTM, YOH	$480 \times 720 \times 120 \times 100$
half-cylinder [35]	VLM: 160, 320, 640, 6400	$640 \times 240 \times 80 \times 100$
ionization [36]	T, PD, H+, H2, He	$600 \times 248 \times 248 \times 100$
Tangaroa [37]	ACC, DIV, VLM, VTM	$300 \times 180 \times 120 \times 150$
five-jet	E	$128 \times 128 \times 128 \times 2000$
tornado [39]	VLM	$128 \times 128 \times 128 \times 48$
vortex [40]	VTM	$128 \times 128 \times 128 \times 90$

Table 2: Average PSNR (dB), LPIPS, and CD values for training across multiple variables and all timesteps. We list the experimented variables and chosen isovalues for computing CD. $u_s = 4$ and $u_t = 3$. The best quality performances are shown in bold.

dataset	method	PSNR ↑	LPIPS ↓	CD ↓
combustion (HR, MF, VTM, YOH) ($v = 0.3, 0.0, 0.0, 0.0$)	TL	29.10	0.195	7.81
	STNet	28.97	0.387	9.65
	CoordNet	34.43	0.165	4.20
	STSR-INR	**34.68**	**0.158**	**3.42**
half-cylinder (VLM: 160, 320, 640, 6400) ($v = 0.0, 0.0, 0.0, 0.0$)	TL	30.79	0.043	5.68
	STNet	36.19	0.028	1.98
	CoordNet	36.87	**0.022**	2.02
	STSR-INR	**38.59**	0.029	**1.77**
ionization (T, PD, H2, H+) ($v = 0.0, 0.0, -0.7, -0.3$)	TL	33.16	0.224	5.76
	STNet	29.72	0.264	10.63
	CoordNet	**40.85**	**0.167**	**1.87**
	STSR-INR	39.62	0.172	3.00

4.2. Baselines and Evaluation Metrics

Baselines. We compare our STSR-INR with three baseline solutions:



(a) TL (b) STNet (c) CoordNet (d) STSR-INR (e) GT

Fig. 5: Super-resolution: comparing volume rendering results. Top to bottom: combustion (YOH), half-cylinder (VLM: 320), and ionization (H+).

Table 3: Total training and inference time (hours) and the model size (MB) for the ionization (T, PD, H2, H+) dataset. $u_s = 4$ and $u_t = 3$.

method	training	inference	model
STNet	25.9	1.5	62.56
CoordNet	19.6	5.5	5.67
STSR-INR	22.9	3.7	5.41

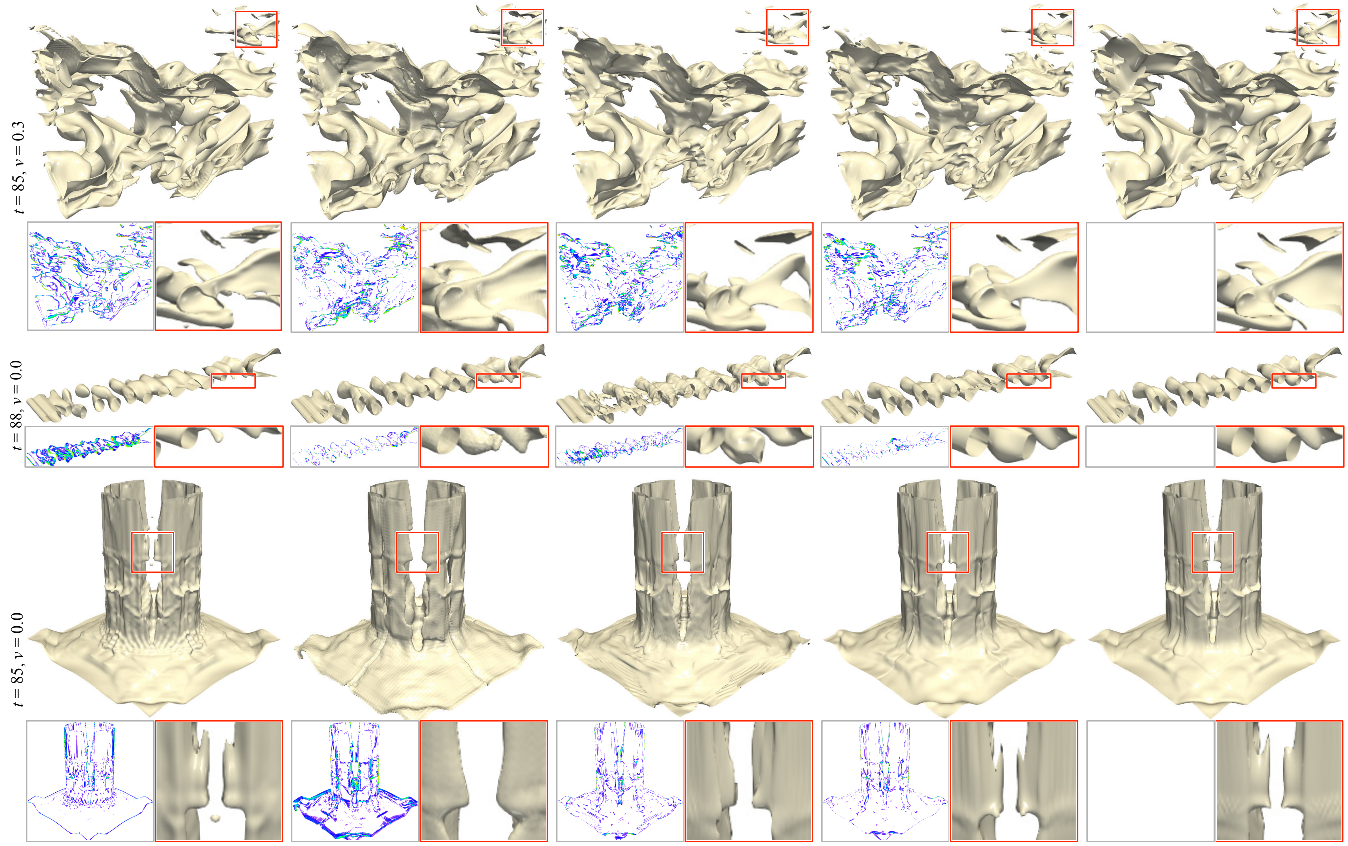
- TL applies tricubic interpolation on the spatial domain, followed by linear interpolation on the temporal domain to upscale volumes in space and time, respectively, to achieve STSR.
- STNet [2] is an end-to-end GAN-based STSR model. We train one STNet on all variables to handle multivariate datasets simultaneously to achieve multivariate STSR learning. The network structure of STNet remains the same, and it does not differentiate variables. We treat volumes of different variables as additional samples for training and inference.
- CoordNet [4] is a general INR network for data generation and visualization generation tasks. For STSR, the network takes spatiotemporal coordinates as input and outputs corresponding voxel values. We modify CoordNet by changing the last output layer from inferring the voxel value of one variable to those of multiple variables.

The training epochs for STNet (including pre-training and fine-tuning) and CoordNet follow the suggestions given in Han et al. [2] and Han and Wang [4], and we empirically found those hyperparameter work well on most datasets. Note that STNet can only upscale the input dataset with fixed spatial and temporal upscale factors (u_s and u_t). It requires low- and high-resolution volume pairs for supervised training. In contrast, CoordNet and STSR-INR support arbitrary u_s and u_t, and they can train the network in an unsupervised manner. However, when training on STSR tasks for multivariate or ensemble datasets, CoordNet can only work with multiple datasets of the same spatiotemporal resolution in the same network. Unlike CoordNet, STSR-INR can train and infer multiple multivariate datasets of different spatiotemporal resolutions in the same network.

Evaluation metrics. We evaluate our reconstruction results based on three metrics. We utilize *peak signal-to-noise ratio* (PSNR) at the data level, *learned perceptual image patch similarity* (LPIPS) [41] at the image level, and *chamfer distance* (CD) [42] at the surface level. The calculation is based on the data, volume rendering images, and isosurfaces coming from the original data and their corresponding version generated from one of the methods.

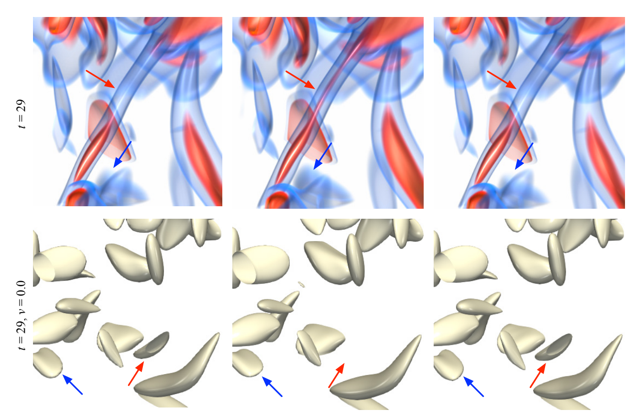
4.3. Results

Quantitative results. Table 2 reports the quantitative results of the four methods across three metrics over three datasets, given the spatial and temporal upscale factors of 4 and 3. STNet performs the worst for the combustion dataset, followed by TL and CoordNet. STSR-INR achieves the best results. TL performs the worst for the half-cylinder (VLM) dataset, while the other three methods get similar results, with STSR-INR yielding the best results for PSNR and CD values. STNet gets the



(a) TL (b) STNet (c) CoordNet (d) STSR-INR (e) GT

Fig. 6: Super-resolution: comparing isosurface rendering results. Top to bottom: combustion (HR), half-cylinder (VLM: 160), and ionization (T).



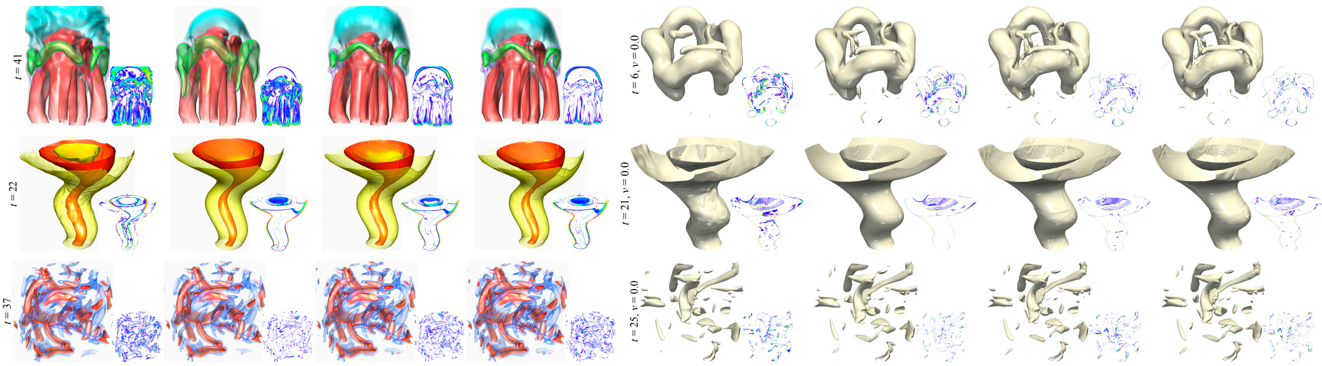
(a) input (b) TL (c) STSR-INR

Fig. 7: Unsupervised training: comparing volume rendering and isosurface rendering of the vortex dataset. $u_s = 1.75$ and $u_t = 2.5$.

worst results for the ionization dataset, followed by TL. CoordNet outperforms STSR-INR. Overall, we can summarize that CoordNet and STSR-INR are the two top methods among these four, and STSR-INR has a slight edge over CoordNet, considering the parameters size of STSR-INR is slightly smaller than CoordNet (as described in Table 3). We attribute STNet's inferior performance to the simultaneous training of multiple variables with various structural appearances, which negatively impacts the discernibility of its temporal discriminator. On the other hand, the interpolation of TL only leverages local neighboring voxel values instead of the global pattern, which leads to a less accurate reconstruction.

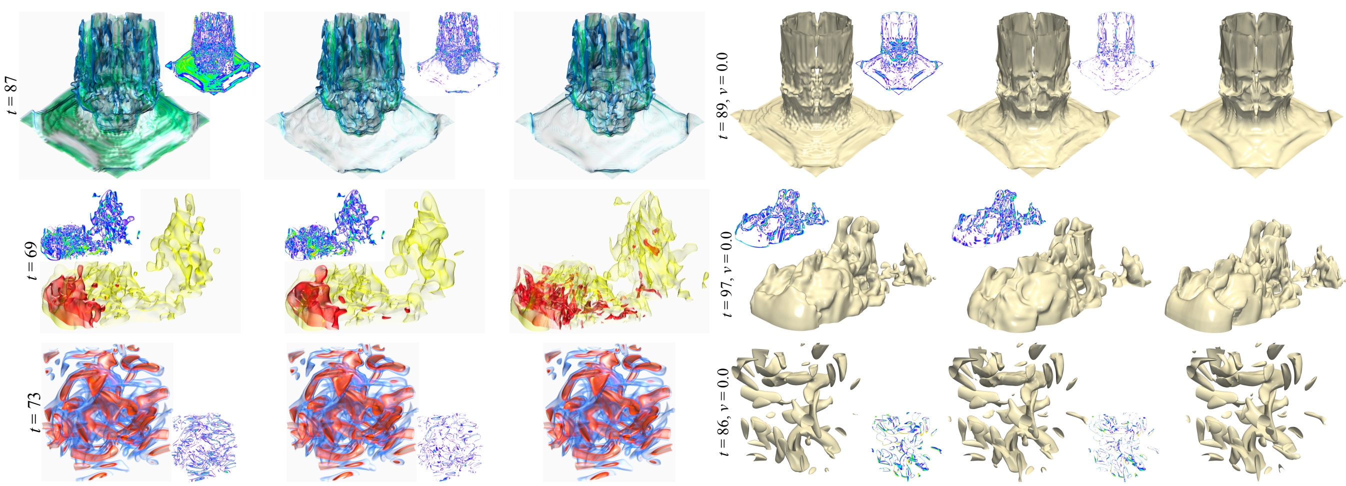
In Figure 3, we plot the three metrics over time averaged across the variables for these three datasets. The periodical rises and falls on each performance curve are due to the setting of temporal upscale factor $u_t = 3$. The timesteps used for training are $\{1, 5, 9, \dots, \}$ (high performance) and the synthesized ones are $\{2, 3, 4, 6, 7, 8, \dots, \}$ (low performance). In Figure 4, we plot the three metrics for each ensemble of the half-cylinder (VLM) dataset. As expected, the higher the Reynolds number, the more turbulent or complex the underlying flow, and the worse the performance.

Qualitative results. Figure 5 shows volume rendering results generated from data produced by these four methods and GT data using combustion (YOH), half-cylinder (VLM: 320), and ionization (T) datasets. To pinpoint the differences, we compute the pixel-wise difference images (i.e., the Euclidean distance in the CIELUV color space) between each method and GT. Noticeable differences are mapped to purple, blue, green, yellow, and red, showing low to high pixel-wise differences (refer to the top-left image of Figure 5 for the colormap legend). In addition, we also highlight a zoom-in region for a closer comparison. These visual comparison results confirm that CoordNet and STSR-INR are better than TL and STNet. Between CoordNet and STSR-INR, STSR-INR leads to rendering results closer to GT renderings.



(a) CoordNet (b) CoordNet* (c) STSR-INR (d) STSR-INR* (e) CoordNet (f) CoordNet* (g) STSR-INR (h) STSR-INR*

Fig. 8: Joint training of datasets with the same resolution: comparing volume rendering and isosurface rendering results for the same spatial and temporal upscale factors. Top to bottom: five-jet, tornado, and vortex. CoordNet* and STSR-INR* denote separate training.



(a) TL (b) STSR-INR (c) GT (d) TL (e) STSR-INR (f) GT

Fig. 9: Joint training of datasets with different resolutions: comparing volume rendering and isosurface rendering results for different spatial and temporal upscale factors. Top to bottom: ionization (He), Tangaroa (VLM), and vortex.

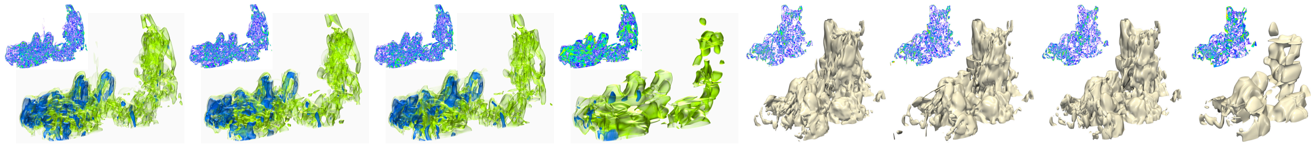
In Figure 6, we show isosurface rendering results generated from these methods using the same datasets as Figure 5 but with different variables. Although the difference images reveal more subtle rendering deviations from GT, the zoom-in regions all indicate that STSR-INR yields isosurfaces most similar to GT while keeping the overall rendering image difference small. CoordNet leads to non-smooth surfaces with clear visual artifacts for the half-cylinder (VLM: 160) dataset. The same can be observed for TL and STNet of the combustion (HR) dataset and TL and CoordNet of the ionization (T) dataset.

Timing and model size. In Table 3, we report the training and inference time of the STSR task and the model size on the ionization dataset for the three deep learning methods: STNet, CoordNet, and STSR-INR. In terms of training, CoordNet achieves the fastest convergence speed. On the other hand, due to the use of multi-head training, STSR-INR takes more time to approximate its optimal parameters for each head (refer to Section 4.5 for an additional performance tradeoff study of STSR-INR with different head settings). STNet requires the most training time because of its larger model size and complicated training pipeline. For inference, the CNN- and LSTM-based STNet requires the shortest time. STSR-INR achieves a faster inference speed than CoordNet as the multi-head structure enables the model to effectively decrease the number of necessary feedforward passes to reconstruct the whole dataset. As for the mode size, STNet has the largest size due to its CNN network structure. CoordNet and STSR-INR have an order of magnitude smaller model size thanks to their MLP structure.

4.4. Unsupervised Training and Joint Training

Unsupervised training. The INR-based solutions allow us to perform unsupervised spatiotemporal super-resolution training where the trained resolution is at the original resolution. We aim to upscale the data to higher-resolution ones with no GT data available for training and comparison. To evaluate unsupervised training of STSR-INR, we compare TL and STSR-INR using the vortex dataset. To demonstrate that STSR-INR permits data upscaling with arbitrary scale factors, we use non-integer upscale factors $u_s = 1.75$ and $u_t = 2.5$, upscaling the original resolution from $128 \times 128 \times 128 \times 90$ to $224 \times 224 \times 224 \times 225$.

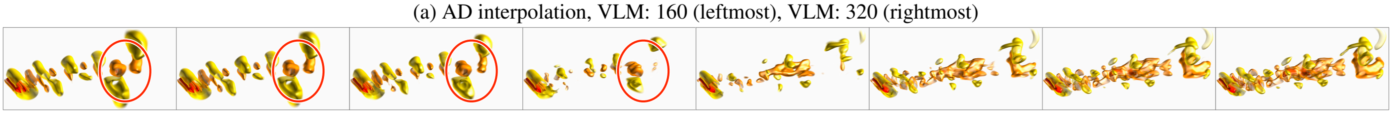
Figure 7 shows the zoomed-in volume rendering and isosurface rendering results. The input low-resolution reveals the jaggy surface boundary in the isosurface rendering. TL and



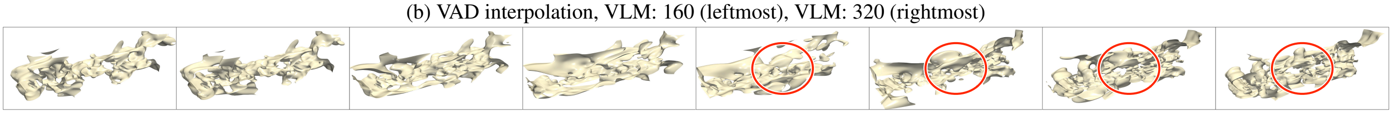
(a) 4-head (b) 8-head (c) 16-head (d) 64-head (e) 4-head (f) 8-head (g) 16-head (h) 64-head

Fig. 10: Multi-head training: comparing volume rendering (VTM, $t = 74$) and isosurface rendering (ACC, $t = 61$, $v = -0.7$) of the Tangaroa dataset.

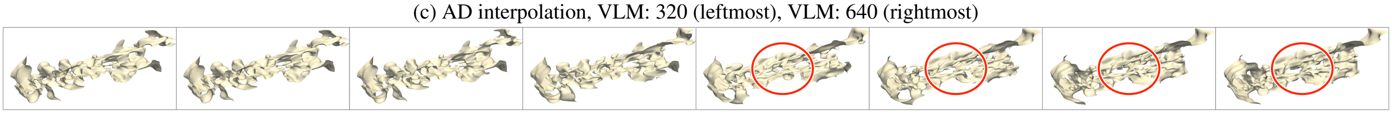
(a) AD interpolation, VLM: 160 (leftmost), VLM: 320 (rightmost)



(b) VAD interpolation, VLM: 160 (leftmost), VLM: 320 (rightmost)



(c) AD interpolation, VLM: 320 (leftmost), VLM: 640 (rightmost)



(d) VAD interpolation, VLM: 320 (leftmost), VLM: 640 (rightmost)

Fig. 11: Latent-space interpolation: comparing volume rendering and isosurface rendering of the half-cylinder (VLM) dataset. $t = 49$ and $v = 0.3$.

Table 4: Average PSNR (dB), LPIPS, and CD values for joint training datasets of the same (top part) and different (bottom part) resolutions. CoordNet* and STSR-INR* denote separate training (only the better ones of CoordNet and STSR-INR are bolded).

dataset	method	PSNR ↑	LPIPS ↓	CD ↓
five-jet ($u_s = 2$, $u_t = 3$, $v = 0.0$)	CoordNet	36.84	0.181	2.65
	CoordNet*	38.46	0.140	1.51
	STSR-INR	**39.33**	**0.103**	**1.08**
	STSR-INR*	39.72	0.079	0.99
tornado ($u_s = 2$, $u_t = 3$, $v = 0.0$)	CoordNet	37.19	**0.081**	1.20
	CoordNet*	39.05	0.076	0.68
	STSR-INR	**40.19**	0.091	**0.54**
	STSR-INR*	39.75	0.092	0.71
vortex ($u_s = 2$, $u_t = 3$, $v = 0.0$)	CoordNet	31.96	0.099	1.26
	CoordNet*	37.25	0.065	0.87
	STSR-INR	**35.25**	**0.076**	**1.04**
	STSR-INR*	37.06	0.070	0.89
ionization (He) ($u_s = 4$, $u_t = 5$, $v = 0.0$)	TL	25.27	0.195	3.03
	STSR-INR	**33.40**	**0.113**	**0.97**
Tangaroa (VLM) ($u_s = 5$, $u_t = 7$, $v = 0.0$)	TL	29.29	**0.124**	4.31
	STSR-INR	**32.23**	0.129	**2.89**
vortex ($u_s = 2$, $u_t = 3$, $v = 0.0$)	TL	30.16	0.090	2.05
	STSR-INR	**37.17**	**0.064**	**0.79**

Table 5: Average PSNR (dB), LPIPS, CD values, model sizes (MB), training time (hours), and speed-ups with STSR-INR for the Tangaroa (ACC, DIV, VLM, VTM) dataset. $u_s = 4$, $u_t = 3$, and $v = (-0.7, -0.9, 0.1, -0.8)$. A similar GPU memory size is used for all these cases during training.

# heads	PSNR ↑	LPIPS ↓	CD ↓	model	training	speed-up
1	**35.91**	**0.124**	**2.31**	5.41	43.75	1×
4	35.55	0.129	2.70	5,41	9.83	4.5×
8	35.56	0.128	2.55	5.41	5.95	7.4×
16	35.58	0.132	2.68	5.41	4.42	9.9×
64	33.11	0.187	5.90	5.41	3.38	12.9×
512	30.29	0.302	15.53	5.53	13.25	3.3×

Table 6: Average PSNR (dB), LPIPS, and CD values with STSR-INR for the half-cylinder (VLM: 160, 320, 640, 6400) dataset. $u_s = 4$, $u_t = 3$, and $v = 0.3$.

scheme	PSNR ↑	LPIPS ↓	CD ↓
AD	38.02	**0.022**	**5.67**
VAD	**38.59**	0.029	6.59

STSR-INR give smooth boundaries. However, as the arrows indicate, the super-resolution results of STSR-INR are closer to input than TL. TL introduces non-existing artifacts (red arrows in the volume rendering images) or misses a surface component (red arrows in the isosurface rendering images) due to its lack of ability to extract the global pattern from low-resolution data. This demonstrates the advantage of STSR-INR over straightforward TL.

Joint training of datasets with the same resolution. Due to their network differences, in each training epoch, CoordNet "sees" all the variables' values at a specific spatiotemporal coordinate while we only allow STSR-INR to update the parameters of one variable. Therefore, CoordNet fits datasets where the variables share similar appearances better than STSR-INR because CoordNet spends less effort identifying the relationship between different variables (refer to the performance results of the ionization dataset shown in Table 2). However, the performance could drop when tackling datasets where the variables exhibit diverse appearances. Still, STSR-INR is robust in handling this scenario. Here, we conduct a comparative study on three variables from different datasets (five-jet, tornado, and vortex), and these variables have no relationship. Each variable has a spatial resolution of $128 \times 128 \times 128$. Because CoordNet can only train and infer on datasets with the same spatiotemporal resolution, we select a subset of timesteps for five-jet and vortex datasets to match the temporal resolution (48 timesteps) of the tornado dataset.

As shown in the top part of Table 4, between CoordNet and STSR-INR, STSR-INR is the winner of eight out of nine metrics across the three variables from different datasets. To better demonstrate the performance drop of CoordNet, we evaluate the performance of separate training for both CoordNet and STSR-INR, denoted as CoordNet* and STSR-INR*. Compared with joint training, STSR-INR and STSR-INR* have a small performance gap for the five-jet and tornado datasets and a drop of 1.81dB in PSNR for the vortex dataset. However, CoordNet suffers a large drop for all three datasets, especially for the vortex dataset, where a drop of 5.29dB in PSNR is reported. Figure 8 shows the rendering results. For volume rendering, STSR-INR yields closer results than CoordNet for five-jet and vortex, only losing to CoordNet at the top region of the tornado while better preserving the overall shape. For isosurface rendering, STSR-INR beats CoordNet for all three datasets. Between STSR-INR and STSR-INR*, STSR-INR produces results of better quality for tornado (volume rendering and isosurface rendering), slightly worse quality for five-jet (isosurface rendering) and vortex (volume rendering and isosurface rendering), and worse quality for five-jet (volume rendering). Again, the visual differences are marginal in all cases. For all three datasets, the visual quality of CoordNet is inferior to that of CoordNet*. By comparing the performance differences between CoordNet and CoordNet*, as well as STSR-INR and STSR-INR*, we observe robust reconstruction of STSR-INR even though the variables exhibit diverse differences. This also suggests that joint training for STSR-INR only incurs slight performance drops, which makes it a feasible alternative to separate training.

Joint training of datasets with different resolutions. Unlike CoordNet and STNet, a significant advantage of STSR-INR is that it permits joint training across multivariate datasets with different spatiotemporal resolutions and upscale factors. We use ionization (He), Tangaroa (VLM), and vortex datasets with varying u_s and u_t to evaluate the joint-training performance of STSR-INR. For network training, the input spatiotemporal resolutions of these datasets are $150 \times 62 \times 62 \times 17$, $60 \times 36 \times 24 \times 19$, and $64 \times 64 \times 64 \times 23$, respectively.

The bottom part of Table 4 shows that STSR-INR beats TL for eight out of nine metrics across the three variables from different datasets. Figure 9 gives the rendering results. For volume rendering, STSR-INR beats TL for ionization (He) and vortex and falls behind TL slightly for Tangaroa (VLM). For isosurface rendering, STSR-INR leads to closer results than TL for ionization (He) and vortex and produces isosurfaces of similar quality for Tangaroa (VLM), while the average CD over all timesteps is still better than TL.

4.5. Network Analysis

To analyze STSR-INR, we conduct network analysis on two key settings: multi-head and VAD. In the appendix, we study the impact of network depth, latent vector length, and ReLU vs. Sine modulator activation function on network performance.

Multi-head analysis. The multi-head design serves as a speed-up option for STSR-INR. However, when implementing this scheme, we need to concatenate the output of different heads to produce the reconstruction results, which could impact the downstream rendering quality. Here, we use the Tangaroa dataset to evaluate the relationship of head numbers with reconstruction quality, model size, and speed-up. We experiment with five head settings (4, 8, 16, 64, and 512). For 8-head, 64-head, and 512-head, we partition the volume along the x, y, and z axes once, twice, and thrice, respectively. For 4-head, we only partition along the x and y axes once, as the z dimension has the lowest resolution. For 16-head, we further partition along the x axis once based on the 8-head partition result because the x dimension has the highest resolution.

Table 5 reports the quantitative results. We observe that the speed-up of STSR-INR does not always increase linearly as the number of increases. The increasing head-branching structure incurs additional memory access costs for each head's output. When the number of heads increases to a large number (64 or 512), the memory access costs can be rather high, which leads to a decline in the speed-up performance. Figure 10 shows the rendering results for selective cases. The difference images are with respect to the 1-head rendering results. The quality remains similar for volume rendering and isosurface rendering with 4-, 8-, and 16-head settings. With 64-head, the results deteriorate tremendously. For quality, speed, and generalization tradeoffs, we recommend the 8-head setting for STSR-INR, and all results for STSR-INR reported in this paper use this setting.

VAD analysis. To validate the effectiveness of VAD, we train the half-cylinder (VLM) dataset with and without using VAD. For the model without VAD, STSR-INR simply uses an AD to optimize the variable-specific latent vectors Φ that are randomly initialized, and there is no sampling process and KLD loss computation. Thus, the training gets easier. After training, we interpolate the optimized latent vectors to produce intermediate volumes.

Even though Table 6 shows that metric-wise, AD and VAD have slight differences, the VAD-interpolated results lead to smoother and more realistic intermediate renderings than those obtained using AD, as shown in Figure 11. Note that the renderings displayed at both ends of the figure are not identical due to the model's use of AD or VAD. The highlighted ellipses indicate that VAD captures the evolution of volumetric and surface components, while AD yields inconsistent interpolation results. This analysis suggests that compared with AD, the training pipeline of VAD preserves more meaningful "semantic" information about the encoded variables. Note that our latent-space interpolation is different from the surrogate model [43, 44] that takes simulation parameters as input. Latent-space interpolation implicitly models the relationship among different variables, which is less powerful and cannot be treated as a replacement for a surrogate model.

4.6. Limitations

Even though STSR-INR can efficiently reconstruct spatiotemporally-resolved multivariate volume sequences with good quality and support latent-space interpolation, it still faces several limitations. First, STSR-INR can utilize a multi-head strategy to speed up the training and inference process, but its inference speed is still slower than the CNN-based STNet method (refer to Table 3). Meanwhile, the multi-head strategy lacks scalability due to the performance drop as the number of heads increases (refer to Table 5). Second, when different variables in the dataset share similar appearances, STSR-INR could struggle to identify the value relationships among them, leading to a lower reconstruction accuracy than CoordNet (refer to the ionization case in Table 2). Third, for datasets with subtle temporal fluctuation (e.g., five-jet), STSR-INR might not capture such temporal variation when u_t is large or happens to match the fluctuation frequency (refer to the accompanying video showing joint training of datasets with the same resolution). We observe this regardless of whether joint training or separate training is employed. Fourth, like CoordNet, STSR-INR performs training and inference on the normalized data and, therefore, cannot recover the data to its original range. This might impede domain scientists' data examination in certain specialized use cases.

5. Conclusions and Future Work

We have presented STSR-INR, a new deep-learning solution for generating simultaneous spatiotemporal super-resolution for multivariate time-varying datasets. Using VAD and a modulated structure, STSR-INR focuses on the variable dimension and supports joint training of variables from datasets with the same or even different spatiotemporal resolutions and upscale factors. This sets STSR-INR apart from state-of-the-art deep learning methods (STNet and CoordNet). We also leverage a multi-head training strategy to significantly boost the training and inference speed of STSR-INR with only a slight downgrade in quality performance. The experimental results show the advantages of STSR-INR over conventional and existing deep-learning-based solutions: it not only achieves the overall best quality performance but also offers the most flexibility regarding arbitrary upscaling, joint training, and unsupervised training.

For future work, we would like to further explore the latent-space interpolation. The VAD analysis reported in Section 4.5 indicates the promise of our solution in synthesizing simulation data from unseen ensemble members. We will verify this with ensemble simulation applications. Moreover, STSR-INR encodes variable information into latent vectors. We can leverage the learned latent vectors to interpret the relationship between different variables. Finally, our current solution only trains one network from scratch at once. Domain scientists usually generate new simulation outputs based on past ones. Thus, it can be efficient if the training on the newly added data can be performed on a previously-trained neural network incrementally.

Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants IIS-1955395, IIS-2101696, OAC-2104158, and the U.S. Department of Energy through grant DE-SC0023145. The authors would like to thank the anonymous reviewers for their insightful comments.

References

[1] Wang, C, Han, J. DL4SciVis: A state-of-the-art survey on deep learning for scientific visualization. IEEE Transactions on Visualization and Computer Graphics 2023;29(8):3714–3733.

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

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

[4] Han, J, Wang, C. CoordNet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. IEEE Transactions on Visualization and Computer Graphics 2023;29(12):4951–4963.

[5] Zhou, Z, Hou, Y, Wang, Q, Chen, G, Lu, J, Tao, Y, et al. Volume upscaling with convolutional neural networks. In: Proceedings of Computer Graphics International. 2017, p. 38:1–38:6.

[6] Han, J, Wang, C. SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization. IEEE Transactions on Visualization and Computer Graphics 2022;28(6):2445–2456.

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

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

[9] Gu, P, Han, J, Chen, DZ, Wang, C. Scalar2Vec: Translating scalar fields to vector fields via deep learning. In: Proceedings of IEEE Pacific Visualization Symposium. 2022, p. 31–40.

[10] Jiao, C, Bi, C, Yang, L. FFEINR: Flow feature-enhanced implicit neural representation for spatio-temporal super-resolution. arXiv preprint arXiv:230812508 2023;.

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

[12] Li, Z, Yang, J, Liu, Z, Yang, X, Jeon, G, Wu, W. Feedback network for image super-resolution. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2019, p. 3862–3871.

[13] Liang, J, Cao, J, Sun, G, Zhang, K, Van Gool, L, Timofte, R. SwinIR: Image restoration using swin transformer. In: Proceedings of IEEE Conference on Computer Vision Workshops. 2021, p. 1833–1844.

[14] Meyer, S, Wang, O, Zimmer, H, Grosse, M, Sorkine-Hornung, A. Phase-based frame interpolation for video. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2015, p. 1410–1418.

[15] Niklaus, S, Mai, L, Liu, F. Video frame interpolation via adaptive separable convolution. In: Proceedings of IEEE International Conference on Computer Vision. 2017, p. 261–270.

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

[17] Mildenhall, B, Srinivasan, PP, Tancik, M, Barron, JT, Ramamoorthi, R, Ng, R. NeRF: Representing scenes as neural radiance fields for view synthesis. In: Proceedings of European Conference on Computer Vision. 2020, p. 405–421.

[18] Sitzmann, V, Martel, JNP, Bergman, AW, Lindell, DB, Wetzstein, G. Implicit neural representations with periodic activation functions. In: Proceedings of Advances in Neural Information Processing Systems. 2020,.

[19] Lu, Y, Jiang, K, Levine, JA, Berger, M. Compressive neural representations of volumetric scalar fields. Computer Graphics Forum 2021;40(3):135–146.

[20] Weiss, S, Hermüller, P, Westermann, R. Fast neural representations for direct volume rendering. Computer Graphics Forum 2022;41(6):196–211.

[21] Han, M, Sane, S, Johnson, CR. Exploratory Lagrangian-based particle tracing using deep learning. Journal of Flow Visualization and Image Processing 2022;29(3):73–96.

[22] Wu, Q, Bauer, D, Doyle, MJ, Ma, KL. Interactive volume visualization via multi-resolution hash encoding based neural representation. IEEE Transactions on Visualization and Computer Graphics 2023;Accepted.

[23] Chen, Z, Chen, Y, Liu, J, Xu, X, Goel, V, Wang, Z, et al. VideoINR: Learning video implicit neural representation for continuous space-time

super-resolution. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2022, p. 2037–2047.

[24] Wu, Q, Li, Y, Sun, Y, Zhou, Y, Wei, H, Yu, J, et al. An arbitrary scale super-resolution approach for 3D MR images via implicit neural representation. IEEE Journal of Biomedical and Health Informatics 2022;27(2):1004–1015.

[25] Park, JJ, Florence, P, Straub, J, Newcombe, R, Lovegrove, S. DeepSDF: Learning continuous signed distance functions for shape representation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2019, p. 165–174.

[26] Li, T, Slavcheva, M, Zollhöfer, M, Green, S, Lassner, C, Kim, C, et al. Neural 3D video synthesis from multi-view video. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2022, p. 5511–5521.

[27] Wu, Q, Bauer, D, Chen, Y, Ma, KL. HyperINR: A fast and predictive hypernetwork for implicit neural representations via knowledge distillation. arXiv preprint arXiv:230404188 2023;.

[28] Mehta, I, Gharbi, M, Barnes, C, Shechtman, E, Ramamoorthi, R, Chandraker, M. Modulated periodic activations for generalizable local functional representations. In: Proceedings of IEEE International Conference on Computer Vision. 2021, p. 14194–14203.

[29] Zadeh, A, Lim, YC, Liang, PP, Morency, LP. Variational auto-decoder: A method for neural generative modeling from incomplete data. arXiv preprint arXiv:190300840 2013;.

[30] Kingma, DP, Welling, M. Auto-encoding variational Bayes. arXiv preprint arXiv:13126114 2013;.

[31] Chen, H, He, B, Wang, H, Ren, Y, Lim, SN, Shrivastava, A. NeRV: Neural representations for videos. In: Proceedings of Advances in Neural Information Processing Systems. 2021, p. 21557–21568.

[32] Aftab, A, Morsali, A, Ghaemmaghami, S. Multi-head ReLU implicit neural representation networks. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing. 2022, p. 2510–2514.

[33] Wang, C, Ma, KL. A statistical approach to volume data quality assessment. IEEE Transactions on Visualization and Computer Graphics 2008;14(3):590–602.

[34] Hawkes, ER, Sankaran, R, Sutherland, JC, Chen, JH. Scalar mixing in direct numerical simulations of temporally evolving plane jet flames with skeletal CO/H2 kinetics. Proceedings of the Combustion Institute 2007;31(1):1633–1640.

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

[36] Whalen, D, Norman, ML. Ionization front instabilities in primordial H II regions. The Astrophysical Journal 2008;673:664–675.

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

[38] Yüce, G, Ortiz-Jiménez, G, Besbinar, B, Frossard, P. A structured dictionary perspective on implicit neural representations. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. 2022, p. 19206–19216.

[39] Crawfis, RA, Max, N. Texture splats for 3D scalar and vector field visualization. In: Proceedings of IEEE Visualization Conference. 1993, p. 261–267.

[40] Silver, D, Wang, X. Tracking and visualizing turbulent 3D features. IEEE Transactions on Visualization and Computer Graphics 1997;3(2):129–141.

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

[42] Barrow, HG, Tenenbaum, JM, Bolles, RC, Wolf, HC. Parametric correspondence and chamfer matching: Two new techniques for image matching. In: Proceedings of International Joint Conference on Artificial Intelligence. 1977, p. 659–663.

[43] He, W, Wang, J, Guo, H, Wang, KC, Shen, HW, Raj, M, et al. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. IEEE Transactions on Visualization and Computer Graphics 2019;26(1):23–33.

[44] Shi, N, Xu, J, Li, H, Guo, H, Woodring, J, Shen, HW. VDL-Surrogate: A view-dependent latent-based model for parameter space exploration of ensemble simulations. IEEE Transactions on Visualization and Computer Graphics 2022;29(1):820–830.

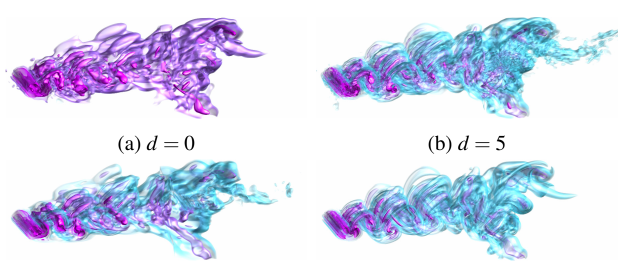
Appendix

Besides the multi-head and VAD analysis investigated in the paper, we study three parameters influencing STSR-INR training: network depth, latent vector length, and modulator activation function.

Network depth. To determine the optimal network depth, namely, the number of residual blocks d for STSR-INR to achieve the best performance, we conduct a parameter study on the half-cylinder (VTM: 640 and 6400) dataset. The results are presented in Table 1. We observe that for multivariate STSR, the network's generalization capability is crucial in achieving high accuracy. Overfitting becomes a significant issue as d increases beyond five, leading to a big drop in accuracy. On the other hand, when there are no residual blocks, the model suffers from underfitting, as shown in Figure 1. To balance the generalization and fitting capabilities. we utilize five residual blocks for our STSR-INR.

Table 1: Average PSNR (dB), LPIPS, and model size (MB) for STSR-INR with different numbers of residual blocks d for the half-cylinder (VTM: 640 and 6400) dataset. $u_s = 4$ and $u_t = 3$.

d	PSNR ↑	LPIPS ↓	model
0	36.69	0.215	0.39
5	**38.43**	0.205	5.41
10	36.53	**0.193**	10.43



(a) $d = 0$ (b) $d = 5$ (c) $d = 10$ (d) GT

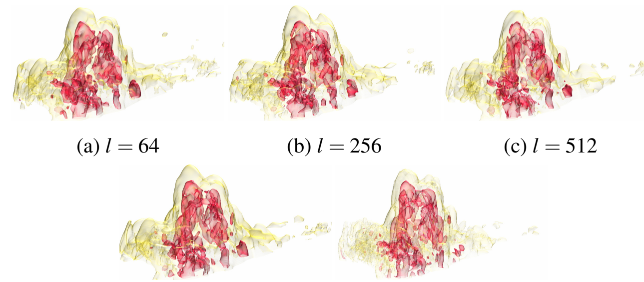
Fig. 1: Network depth: comparing volume rendering of half-cylinder (VTM: 640) dataset. $u_s = 4$ and $u_t = 3$.

Latent vector length. To assess the impact of the latent vector length l, we conduct a parameter study on the Tangaroa (VLM and ACC) dataset. The results are summarized in Table 2. We observe that varying l does not significantly influence the reconstruction result, as shown in Figure 2. Although $l = 1024$ achieves the best result, we find that a length of 256 achieves a similar level of accuracy while utilizing fewer parameters. Therefore, we choose $l = 256$ for our STSR-INR.

Table 2: Average PSNR (dB) and LPIPS for STSR-INR with different latent vector lengths for the Tangaroa (VLM and ACC) dataset. $u_s = 5$ and $u_t = 3$.

latent vector length	PSNR ↑	LPIPS ↓
64	33.25	0.201
256	33.29	**0.199**
512	33.27	0.211
1024	**33.40**	0.226

Modulator activation function. Mehta et al. [1] applied the ReLU activation function for their modulator network. Nevertheless, applying a Sine activation to the modulator network could stabilize network training as the input and output ranges



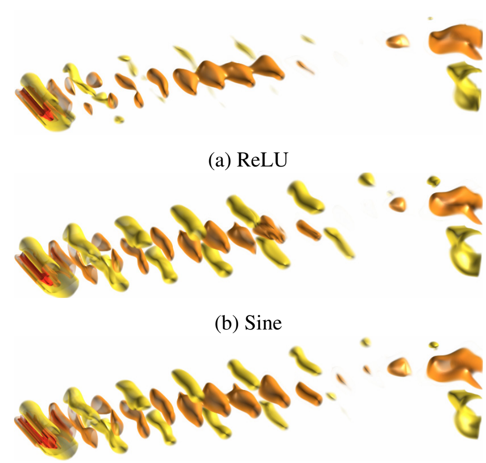
(a) $l = 64$ (b) $l = 256$ (c) $l = 512$ (d) $l = 1024$ (e) GT

Fig. 2: Latent vector length: comparing volume rendering of the Tangaroa (VLM) dataset. $u_s = 5$ and $u_t = 3$.

in the synthesis network remain [-1, 1]. We conduct a comparative study on the ReLU or Sine modulator activation function to demonstrate this. The results shown in Table 3 and Figure 3 suggest that the modulator with Sine activation can achieve a significantly higher reconstruction accuracy than ReLU.

Table 3: Average PSNR (dB) and LPIPS for STSR-INR with different modulator activation functions for the half-cylinder (VTM: 160 and 320). $u_s = 4$ and $u_t = 3$.

activation function	PSNR ↑	LPIPS ↓
ReLU	27.08	0.039
Sine	**41.11**	**0.015**



(a) ReLU (b) Sine (c) GT

Fig. 3: Modulator activation function: comparing volume rendering of half-cylinder (VLM: 160) dataset. $u_s = 4$ and $u_t = 3$.

References

[1] Mehta, I, Gharbi, M, Barnes, C, Shechtman, E, Ramamoorthi, R, Chandraker, M. Modulated periodic activations for generalizable local functional representations. In: Proceedings of IEEE International Conference on Computer Vision. 2021, p. 14194–14203.