# Deep Learning-based Upscaling for In Situ Volume Visualization

Sebastian Weiss, Jun Han, Chaoli Wang, and Rüdiger Westermann

**Abstract** Complementary to the classical use of feature-based condensation and temporal subsampling for in situ visualization, learning-based data upscaling has recently emerged as an interesting approach that can supplement existing in situ volume visualization techniques. By upscaling we mean the spatial or temporal reconstruction of a signal from a reduced representation that requires less memory to store and sometimes even less time to generate. The concrete tasks where upscaling has been shown to work effectively are geometry upscaling, to infer high-resolution geometry images from given low-resolution images of sampled features; upscaling in the data domain, to infer the original spatial resolution of a 3D dataset from a downscaled version; and upscaling of temporally sparse volume sequences, to generate refined temporal features. In this book chapter, we aim at providing a summary of existing learning-based upscaling approaches and a discussion of possible use cases for in situ volume visualization. We discuss the basic foundation of learning-based upscaling, and review existing works in image and video super-resolution from other fields. We then show the specific adaptations and extensions that have been proposed in visualization to realize upscaling tasks beyond color images, discuss how these approaches can be employed for in situ visualization, and provide an outlook on future developments in the field.

## 1 Introduction

For in situ volume visualization, two commonly employed approaches are feature-based data reduction and spatial or temporal subsampling. In the former approach,

Sebastian Weiss, Rüdiger Westermann
Technical University of Munich, Garching bei München, Germany, e-mail: `{sebastian13.weiss,westermann}@tum.de`

Jun Han, Chaoli Wang
University of Notre Dame, Notre Dame, Indiana, USA, e-mail: `{jhan5,chaoli.wang}@nd.edu`

each dataset is condensed in situ to a few important features, and these features are used to analyze the data. Since extracted features usually require far less memory than the original data, memory bandwidth and capacity limitations can be overcome. On the other hand, features not selected are lost, and some feature extraction techniques require global data access operations not well supported by the data distribution scheme on the parallel computing architecture. In the later approach, volumetric data are downscaled spatially or only every $n$-th timestep is stored, thus reducing the amount of memory needed. Then, spatial or temporal features can get lost, since classical interpolation schemes cannot reconstruct these features from the reduced data in general.

Complementary to these approaches, learning-based data upscaling has recently emerged as an interesting approach that can supplement existing in situ volume visualization techniques. By *upscaling* (which is also referred to as *super-resolution*), we refer here to the spatial or temporal reconstruction of a physical field from a reduced representation. The concrete tasks where learning-based upscaling has been demonstrated for scientific data are

1 *Image upscaling*: Upscaling in the visualization image domain by inferring high-resolution images from given low-resolution images of sampled 3D features.
2 *Spatial upscaling*: Upscaling in the spatial domain by inferring a higher resolution of a 3D dataset from a downscaled version for reconstructing spatial features.
3 *Temporal upscaling*: Upscaling in the temporal domain by inferring temporally dense volume sequences from sparse sequences for refining temporal features.

Option (1) requires focusing on specific features that are visualized at low image resolution. Yet, resulting images require less data access operations to generate, can be generated at higher speed, and decrease the memory required to store and transmit them, e.g., for use over low bandwidth channels in remote visualization environments. With options (2) and (3), storing a dataset becomes faster, and more datasets can be stored in a given a certain disk capacity. Depending on the size of the downsampled dataset, it can be even feasible to stream an entire dataset to a remote client where the visualization occurs. This can be an interesting option to monitor the running simulation.

The challenge in upscaling is to infer the structure of a coarsely sampled dataset from a low-resolution spatial or temporal sampling, beyond what can be predicted from the given samples by classical upscaling filters like bilinear or bicubic interpolation. In general, this seems impossible without any further assumptions about the structures that are contained in the dataset. Recent works in learning with artificial neural networks (ANNs), however, have demonstrated that such networks have the capabilities to learn such assumptions.

In abstract terms, upscaling seeks a mapping function from inputs to outputs, called the generator. In the in situ scenario, the generator learns to map a multi-dimensional field, e.g., a 2D color or geometry image, or a 2D or 3D and possibly time-varying scalar or vector field, to a higher spatial or temporal resolution, optionally including additional parameters, i.e., channels, that are inferred from the given input samples.

With ANNs, the generator internally builds a so-called *latent space representation* that encodes the nonlinear mapping function. The generator tries to learn an identity mapping, which gives for every low-resolution input the corresponding high-resolution output. Since the dimension of the latent space is not sufficient to achieve such a mapping for every input, the network learns to encode relevant features that have a significant impact on the inference quality. Thus, the generator learns assumptions about the occurrence of structures by using corresponding pairs of low- and high-resolution fields in the training process. Learned assumptions can then be transferred to a new low-resolution input, to generate a high-resolution variant that adheres to the structures seen at training time.

Learning-based image and video super-resolution have achieved remarkable results, by training networks using corresponding pairs of low- and high-resolution color images [9, 26]. Learned assumptions can then be transferred to a new low-resolution input, to generate a high-resolution variant in which structures that have been seen at training time are well recovered. Similar approaches have been used in the context of numerical fluid simulation, in particular, to add turbulent sub-structures to low-resolution input simulations [6, 57].

In this book chapter, we aim to bring the readers' attention to the possibilities of learning-based upscaling in the context of in situ volume visualization. Even though learning-based upscaling has not yet been integrated into existing in situ visualization systems, especially in this context, we see it as an interesting technique that can effectively complement existing approaches. In particular, recent works on spatial and temporal upscaling of physical fields indicate that networks do not learn a specific field, but rather that networks can generalize and learn properties of structures that occur in such fields.

On the other hand, as recent works have shown, a network cannot infer the missing data samples accurately in all but very simple scenarios. In particular, when structures appear which have never be seen by the network in the training process, or the sampling frequency is so low that structures are missed entirely, an accurate reconstruction cannot be expected in general. While some networks tend to hallucinate new structures in this case, other network variants prefer a smooth continuation similar to classical smoothing filters.

One way to address this problem is to build specialized networks for certain types of simulations to learn the specific features (and their spatial and temporal relationships) that can occur. To achieve this, further research is required to explore the limits of predictability using network-based inference, including in particular a thorough investigation of their specialization capabilities in different application areas. In this context, it will be important to shed light on the use of additional information sources, besides low-resolution versions of the data, that can be generated efficiently in a running simulation and can help to improve the prediction. For instance, to investigate whether certain feature indicators can be derived and stored together with the low-resolution version, so that the inference step can learn to combine both sources in a meaningful way and improve the inference results.

Another limitation of current network architectures for upscaling tasks is performance, which is essentially bound by the many data access operations a network needs to perform to reconstruct a full resolution dataset from the low-resolution version. We are confident that with faster deep-learning hardware and performance-optimized network architectures the performance and scalability of upscaling networks will increase significantly over the next years. Furthermore, in recent work it has been shown that ANNs can even learn the importance of samples for data reconstruction [54], by training an adaptive sampling network and a reconstruction network end-to-end. This work, in particular, gives evidence that networks can be trained to convert a dataset into a sparse, yet feature-preserving representation from which another network has learned to generate a visualization in turn, i.e., without having to reconstruct the full resolution field. Such an approach can significantly reduce the number of data access operations and, thus, increase the visualization performance accordingly.

## 2 Background and Related Work

### 2.1 Artificial Neural Networks

In the following, we will briefly discuss some of the basic machine learning mechanisms underlying neural network-based upscaling, together with a short description of the specific architectures used in upscaling for visualization. For a thorough introduction to and an exhaustive summary of the developments in neural network-based learning, we refer readers to the overviews by Schmidthuber [45] and Goodfellow et al. [13].

In recent years, ANNs have gained tremendous attention due to their superior performance to alternative methods in many pattern recognition and machine learning tasks. The power of these networks comes from their ability to form representations of categories by “learning” from large sets of samples in which these categories are present. The learned representations are then used to recognize, classify and infer about properties of objects and events in unknown samples. An ANN aims to solve problems in a similar way as a human brain. It consists of interconnected nodes akin to the vast network of neurons in the brain. Artificial neurons are aggregated into layers, and different layers perform different kinds of transformations on their inputs. A *deep neural network* (DNN) is an ANN with multiple layers between the input and output layers [32]. These extra hidden layers result in a *deep* network, enabling the composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing *shallow* network [2]. As an essential branch of machine learning, deep learning has been successfully applied to many applications such as computer vision, speech recognition, natural language processing, and computer graphics, achieving results comparable or even superior to human experts. In recent years, fully connected network layers were replaced by deep hierarchies using only local convolutional update operations, and massively parallel graphics processing units (GPU) equipped with high-performance memory interfaces made learning with large datasets and many layers practicable. *Convolu-*

*tional neural networks* (CNNs) have proven very successful for *supervised* image recognition and classification. The hidden layers of a CNN include several stages of local convolutional and pooling layers, sometimes followed by one or several fully-connected layers. A neuron $z_j^{(i+1)}$ at *convolutional layer* $i + 1$ detects local combinations of features by aggregating information from the previous layer $i$ via convolution operations and non-linear state updates:

$$a_j = W_j * z^i + b_j, \tag{1}$$

$$z_j^{(i+1)} = f(a_j), \tag{2}$$

where $W_j$ is a linear convolution operator with trainable weights, and $b_j$ is a bias used to allow for absolute shifts in the output values. A non-linear function $f$ is used to compute a neuron's final state, it's so-called *activation*, to enable the network to perform a non-linear mapping of its inputs without affecting the receptive fields of the convolutional layers. Typical activation functions are $\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$ or $\text{ReLU}(x) = \max(0, x)$. A *pooling layer* aggregates features into smaller and smaller feature maps via down sampling, i.e., it merges semantically similar features into one by computing a summary (e.g., maximum or average). The *fully-connected layer* learns global features from local ones by connecting neurons to all activations in the previous layer. This is used, for instance, in classification tasks to generate the final classification mask from the activations of all neurons in the final convolutional layer. Today, deep learning [30, 48] refers to the use of high-throughput architectures for learning hierarchical representations of categories, which are formed via non-linear models for transforming the representation at one level into a more abstract one.

For network learning, a *loss function* is used to measure the difference between the predicted network output and the desired result, e.g., a class label in classification tasks or a high-resolution ground truth image in image-based upscaling. Minimizing the loss function is usually achieved with an optimization method such as *stochastic gradient descent* (SGD). SGD first calculates the gradient of the loss function with respect to the network's weights and biases, and then updates the parameters according to a given *learning rate*. The network's weights and biases are the only parameters that can be modified to make the loss as low as possible. Because the loss is calculated through the composition of the neurons' activation functions, it is a continuous and piecewise differentiable function of the parameters. This gives rise to minimization via an iterative process of gradient descent. As such, the learning process reduces to calculating the gradient of a network function with respect to its weights. This is what is performed in the training step.

A simple yet often surprisingly effective class of loss functions considers regular vector norms, i.e., $L_1$ or $L_2$, over the data domain. *Perceptual losses* [12, 10, 25] have been widely adopted in image and video upscaling to guide networks towards additional image details instead of smoothed mean values. The idea is that two images are similar if they have similar activations in the latent space of a trained network. Let $\phi$ be the function that extracts the layer activations when feeding an input image into the network, and let $O^{EST}$ and $O^{GT}$, respectively, be the inferred output and the ground-truth image. The perceptual loss function aims at minimizing the distance

$$\mathcal{L}_P = \sum_s ||\phi(O_s^{GT}) - \phi(O_s^{EST})||_2^2. \tag{3}$$

*Autoencoders* [29] can learn latent features from data in an *unsupervised* manner, e.g., to use these features for dimensionality reduction. An autoencoder consists of two networks: an *encoder* and a *decoder*. The encoder encodes an input data sample to a compressed representation in the latent space. The decoder decodes the latent representation back to reconstruct the data sample as close as possible. *Generative adversarial networks* (GAN) [14] are explicitly designed to optimize for generative tasks. A GAN consists of two networks: a *generator* and a *discriminator*, which contest with each other in a zero-sum game. The generator maps from a latent space to a particular data distribution of interest. The discriminator discriminates between instances from the true data distribution and candidates produced by the generator. In adversarial training, this discriminator is then used in the loss function of the generator network. A popular loss function in adversarial training is the *binary cross-entropy loss*. Let $z$ be the input over all timesteps and $G(z)$ the generated results, and let $D(x)$ be the discriminator that takes the generated results as input and produces a single scalar score. Then the discriminator is trained to distinguish fake from real structures by minimizing

$$\mathcal{L}_{\text{GAN},D} = -\log(D(x)) - \log(1 - D(G(z))). \tag{4}$$

The generator is trained to minimize

$$\mathcal{L}_{\text{GAN},G} = -\log(D(G(z))). \tag{5}$$

*Recurrent neural networks* (RNN) [42, 15] can capture the dynamics of a sequence or a time series and are widely used in speech recognition, text synthesis, and handwriting recognition. As the most popular deep RNN architecture, the *long short-term memory* (LSTM) [22] addresses the vanishing gradient problem in traditional RNNs by adding forget gates in the units. To strengthen the temporal coherence of time series predictions, one can use loss functions that explicitly penalize differences between predictions and different timesteps. To cope with motion in images, let $W_{i,j}$ be a warping operator that aligns the output at time $i$ with the output at time $j$ with respect to some motion field. Then, the temporal loss can be defined as

$$\mathcal{L}_{\text{temp}} = \sum_t ||O^{EST_t} - W_{t-1,t}(O^{EST_{t-1}})||_2^2. \tag{6}$$

## 2.2 Related Work in Upscaling

**Upscaling Image and Video Data.** In recent years, deep learning approaches have been used successfully for single-image super-resolution tasks [8, 47, 49, 31, 51], i.e., the upscaling of images and videos from a lower to some higher resolution. Many previous works let the networks learn to optimize for losses between the inferred and ground-truth images based on direct vector norms [27, 28]. GANs were introduced to prevent the undesirable smoothing of direct loss formulations [43, 33, 43], and instead use a second discriminator network that discriminates real from generated samples and guides the generator. Convolutional architectures [8] with residual blocks [21] are popular generator architectures that offer training stability as well as high-quality inference. Losses based on the feature-space differences of image classification networks, e.g., a pre-trained VGG network [25] or the discriminator in a GAN setting, have shown to mimic well the human's capability to assess the perceptual similarity between two images. In the context of synthetic images, existing work focuses on enhancing path-traced images. A common application is image denoising, e.g., for Monte-Carlo raytracing [58, 4].

In contrast to image super-resolution, video super-resolution tasks introduce the time dimension, and as such, require temporal coherence and consistent image content. While many methods use multiple low-resolution frames [50, 34, 24], the FRVSR-Net [44] reuses the previously generated high-resolution image to achieve better temporal coherence. By using a spatiotemporal discriminator, the Teco-GAN [7] network produced results with spatial detail without sacrificing temporal coherence. Multiframe approaches benefit from aligning the frames via warping, which requires an estimation of the image-space motions. As this is usually not readily available for natural videos, optical flow estimation networks are a popular choice [3]. EDVR [52] uses alignment modules on a U-Net [41] to align features of different frames. Recurrent architectures employ feedback loops to address sequence, time, and video prediction tasks [5]. Chaitanya et al. [4] used recurrent connections to propagate a latent state over time inside the network, while other methods use the previously generated high-resolution output as input [44].

**Upscaling Scientific Data.** Thanks to the tremendous advances of deep learning solutions, visualization researchers have begun to explore the capabilities of DNNs for upscaling and reconstruction of 2D/3D steady and time-varying scientific data, including both scalar and vector fields. Closely related to scientific visualization, Zhou et al. [60] presented a CNN-based solution that upscales a volumetric dataset using three hidden layers designed for feature extraction, non-linear mapping, and reconstruction, respectively. Han et al. [17] took a two-stage approach for vector field reconstruction via deep learning. The first stage initializes a low-resolution vector field based on the input streamline set. The second stage refines the low-resolution vector field to a high-resolution one via CNN. The use of neural network-based inference of data samples in the context of in situ visualization was demonstrated by Han and Wang [18], by letting networks learn to infer missing timesteps between 3D simulation results. Guo et al. [16] designed a deep learning framework that produces coherent spatial super-resolution of 3D vector field data. With a recommended scaling factor of 4 or 8, they can downsample vector field data by 64 or 512 times at simulation time and upsample these reduced data back to their original resolution with good quality.

Beyond scientific visualization, several works aim to upscale physical fields resulting from volumetric flow simulations. For example, Xie et al. [57] presented tempoGAN to synthesize spatial super-resolution volume sequences. Temporal coherence is ensured by wrapping velocity and vorticity fields into the synthesized volumes. Xiao et al. [56] proposed a CNN-based flow correction method for a fast preview of the smoke animation results based on low-resolution simulations, which was achieved through the use of a grid-layer feature vector along with a special loss function. Werhahn et al. [55] designed a multi-pass method to upsample 3D spatiotemporal functions with GANs. They decomposed generative problems on the Cartesian field functions into multiple smaller subproblems for efficient learning. Bai et al. [1] proposed a dynamic upsampling approach to generate high-resolution turbulent smoke flows using a dictionary-based neural network.

## 3 Upscaling Scenarios - Image-based Upscaling

Rendering an accurate image of a volumetric field typically requires a large number of data samples, and reducing this number lies at the core of research in volume rendering. In the following, we shed light on the use of CNNs for learning the upscaling of a low-resolution rendering of an isosurface to a higher resolution [53], with reconstruction of spatial detail and shading. What makes this approach interesting for in situ visualization is the reduced number of data samples that need to be accessed to obtain the final high-resolution image. Since the high-resolution image is inferred from a low-resolution image with only 1/16 (about 6%) of the pixels in the high-resolution version, the number of data access operations that are required to generate the high-resolution image decreases correspondingly. Due to this, in in situ environments where a dataset is so large that volume rendering must be performed while the data is being generated, increasing rendering performance can be expected. In addition, the low-resolution image can be streamed in turn to a remote visualization client, reducing bandwidth requirements by 16:1 and requiring no additional compression stage on the machine where the data is being generated.

The input and output of the image-based upscaling (IU) network in [53] are 2D images of an isosurface in a 3D scalar field. In contrast to classical image- and video-upscaling approaches, however, there are differences in what these images represent and what the network learns to infer. First, the upscaling task does not work on color images. Instead, it uses a geometry image of the isosurface, including depth and gradient information. As illustrated in Figure 1, this leads to improved learning of geometric surface properties and avoids color bleeding effects when different color maps are used in the training process. Second, instead of learning a mapping from a low-pass filtered version of a given high-resolution image to that image, the network is trained to learn a mapping from a low-resolution sampling of the isosurface in the high-resolution dataset to a version that is sampled at high image resolution. Third, the network learns to generate additional attribute channels from the given inputs. In particular, it infers global illumination values from the given geometry images

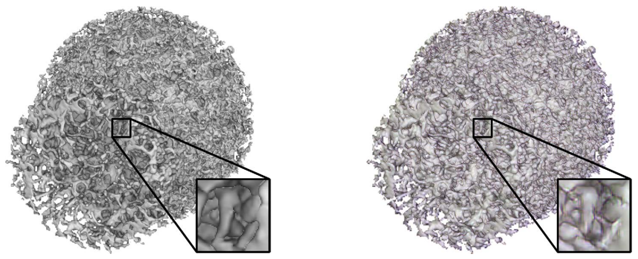

Fig. 1: Left: IU using depth and normal maps with screen-space shading. Right: upscaling of color images introduces color bleeding from color mappings seen during training. This image is adapted from our previous work [53].

by learning the relation between low-resolution geometry and global illumination in the high-resolution data.

**Processing Pipeline.** In every frame $t$, the network receives the low-resolution input $I_t^{LR}$, which comprises multiple 2D fields, the so-called channels. The binary mask $M_t^{LR} \in \{-1,+1\}^{H\times W}$ indicates for every pixel whether or not the isosurface is hit. The normal map $N_t^{LR} \in [-1,+1]^{3\times H\times W}$ stores the the screen-space normal vectors at the rendered surface points. The depth map $D_t^{LR} \in [0,1]^{H\times W}$ stores the distance of each point to the viewing plane.

From these input fields, the network infers the high-resolution image of the isosurface $O_t^{EST}$, including additional output channels that were not given in the low-resolution input, such as high-resolution ambient occlusion (AO) values. Therefore, in the training process, the network is fed with ground-truth AO maps (generated via volumetric raycasting) and learns to predict them according to the low-resolution geometry. Screen-space Phong shading with AO is finally applied as a post-processing step. Optionally, a map of 2D displacement vectors $F_t^{LR} \in [-1,+1]^{2\times H\times W}$ – indicating the screen-space flow from the previous view to the current view – is computed internally to let the network smooth the differences between subsequent predictions.

**Architecture Details.** The network architecture builds upon a fully convolutional frame-recurrent neural network (FRVSR-Net) consisting of a series of residual blocks [44]. In a residual network, some layers feed their output not only into the next layer but also directly into the layers many stages away. By using these so-called skip connections, the layers can learn the residual between the true output and the prediction. Furthermore, larger gradients can be propagated to initial layers, thus avoiding the problem of vanishing gradients in deep networks. The FRVSR-Net is a rather small network that provides a good tradeoff between quality and inference speed, i.e., the inference of FullHD images at 60fps is possible, which is difficult to achieve with more complex architectures like the U-Net [41]. An illustration of the network's building blocks and its topology is given in Figure 2.

The residual architecture network improves the network's capability to generalize to new data, as it can focus on generating the residual content [21]. Hence, the input channels are bilinearly upsampled and added to the channels of the output,

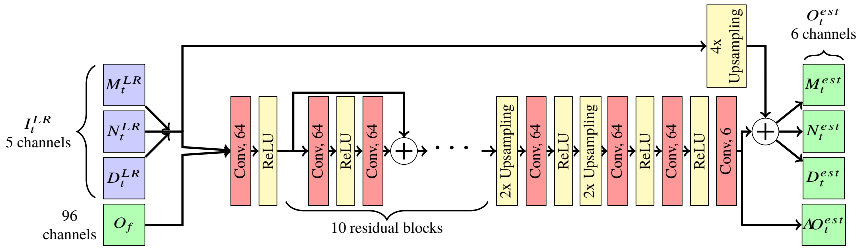

Fig. 2: The IU network architecture. ⊕ indicates component-wise addition of the residual. All convolutions use $3 \times 3$ kernels with stride 1. Bilinear interpolation is used for the upsampling layers. This image is reprinted from our previous work [53].

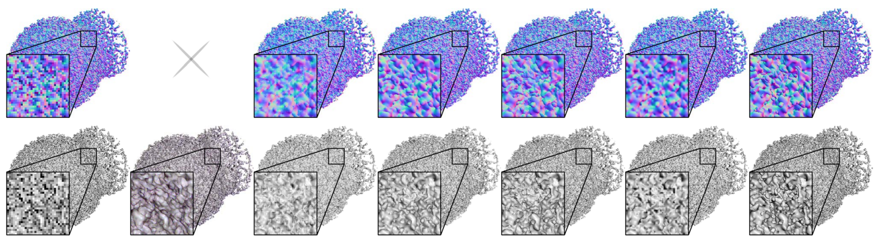

Fig. 3: Comparing networks with different loss function configurations. Top row: the normal map. Bottom row: the shaded output color. Since the "Shaded" network acts directly on color images, a normal map is not used. Left to right: low-resolution input, shaded, $L_1$-color, $L_1$-geometry (final), perceptual, GAN, and ground truth. This image is adapted from our previous work [53].

producing $M_t^{EST}$, $N_t^{EST}$, and $D_t^{EST}$. The only exception is $AO_t^{EST}$, which is inferred from scratch, as there is no low-resolution input AO map.

**Training and Loss Function Characteristics.** For training and validation, images of isosurfaces in a supernova simulation (dataset Ejecta) on a $256^3$ grid are used. The dataset is rendered at different timesteps to provide the network with a variety of different geometric structures, ranging from very small details to rather smooth low-frequency parts. The input images were subdivided into smaller parts so that multiple inputs can be processed at once and benefit from batch processing in the optimizer.

Weiss et al. [53] compare networks with different weighted combinations of the individual losses described in Section 2.2. Figure 3 shows a visual comparison of the surface structures (without AO) that are inferred from the low-resolution input image using these networks.

The evaluation indicates that a network which is trained only with $L_1$-losses and a minor objective on temporal coherence gives the best results. This network only sees shaded colors in the temporal coherence loss during the training process and is thus forced to focus primarily on the reconstruction of geometry. Adding a perceptual loss on the normal and AO fields does not lead to any visual differences. This is because

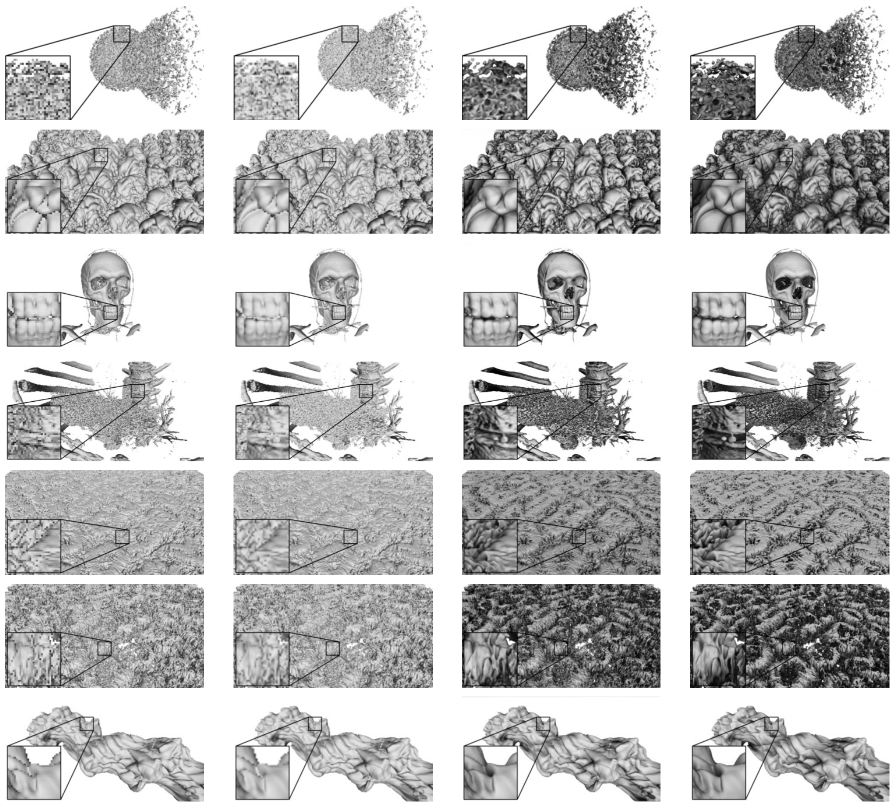
Fig. 4: Comparing upscaling quality on Ejecta, RM, Skull, Thorax, two isosurfaces of a Rayleigh-Bernard process and a single jet. The network was trained only on Ejecta, but on different views than shown. The other datasets were never seen during training. Left to right: input, bilinear, IU, and ground truth with AO. This image is adapted from our previous work [53].

the perceptual loss network VGG-19 is trained on color images and does not explicitly consider the relation between geometry and shaded output. Further experiments using an adversarial loss indicate that the GAN produces more high-frequency details that actually decrease the quality of the reconstruction. Furthermore, it significantly increases both training time and memory requirements of the discriminator.

**Quality and Performance Evaluation.** The following comparison sheds light on the quality and performance of network-based upscaling. The comparison involves images of isosurfaces in Ejecta that were never seen during training, as well as images of isosurface in other datasets (see Figure 4): A numerical simulation of a Richtmyer-Meshkov (RM) instability at $1024^3$ and of a Rayleigh-Bernard process at $1200x1200x80$, CT scans of human anatomies (Skull, Thorax) at a resolution of $256^3$, and a flow simulation (Flow) at $256^3$. For all datasets, even though some of

Table 1: Comparing average PSNR and SSIM values of the normal map for different methods.

| | PSNR (dB) | | | | SSIM | | | |
|---|---|---|---|---|---|---|---|---|
| dataset | nearest | linear | cubic | IU | nearest | linear | cubic | IU |
| Clouds | 59.18 | 65.51 | 56.03 | **69.88** | 0.92 | 0.94 | 0.88 | **0.96** |
| Ejecta | 60.71 | 64.99 | 58.03 | **69.43** | 0.91 | 0.92 | 0.87 | **0.95** |
| RM | 25.21 | 27.26 | 24.02 | **28.50** | 0.74 | 0.76 | 0.69 | **0.80** |
| Thorax | 43.76 | 46.74 | 41.69 | **49.36** | 0.75 | 0.76 | 0.68 | **0.78** |
| Skull | 25.46 | 27.36 | 24.55 | **29.10** | 0.91 | 0.92 | 0.87 | **0.95** |

them exhibit geometric features that are different from those in Ejecta, the inference results are very close to the ground truth. The results indicate that the network can effectively infer meaningful details in line with the geometric surface properties, and can furthermore predict a highly accurate distribution of AO values from the inferred geometry.

The quantitative assessment of the quality of IU using the peak signal-to-noise ratio (PSNR) and the structural similarity index (SSIM) on the normal map in Table 1 confirm the high reconstruction fidelity of network-based inference.

Compared to volumetric raycasting on the GPU without the simulation of AO values, the performance of isosurface image upscaling typically ranges from equal to a factor of 2 faster, including the inference of AO values. Once the renderer also needs to simulate AO, it is outperformed by the upscaling network by two orders of magnitude, due to the computational complexity of AO simulation using ray-based approaches.

## 4 Upscaling Scenarios - 3D Spatial Upscaling

Besides upscaling in image space, spatial upscaling (SU) can also be performed directly in the data space. A possible integration with in situ implementation is as follows. The simulation first outputs, for example, 40% of samples from the entire volume sequence as high-resolution volumes, as the training data. The simulation then resumes and only outputs subsequent low-resolution volumes, as the testing data. For network training, the corresponding low-resolution volumes can be obtained by downsampling the high-resolution ones using the bicubic kernel with a downscaling factor of four (thus, each volume is 1/64 of the original size). Once trained offline, the network can perform real-time inference to predict high-resolution volumes from low-resolution ones.

Given a time-varying volumetric dataset, Han and Wang [19] designed a network architecture similar to tempoGAN [57] for SU of each volume in the sequence. The network includes a generator $G$ and two discriminators (spatial discriminator $D_s$ and temporal discriminator $D_t$). With this architecture, the network produces

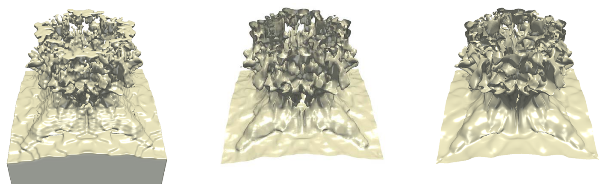

Fig. 5: Comparing same-variable inference results of the Ionization (He+) dataset at timestep 73 using isosurface rendering. The chosen isovalue is -0.84 (the value range is normalized to [-1, 1]). Left to right: bicubic, SU, and ground truth.

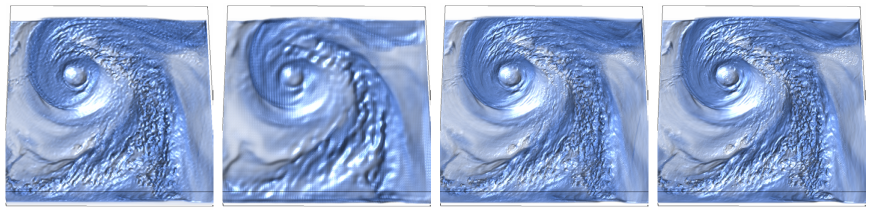

Fig. 6: Comparing same-variable inference results of the Hurricane (QVAPOR) dataset using volume rendering. Left to right: bicubic, CNN, SU, and ground truth.

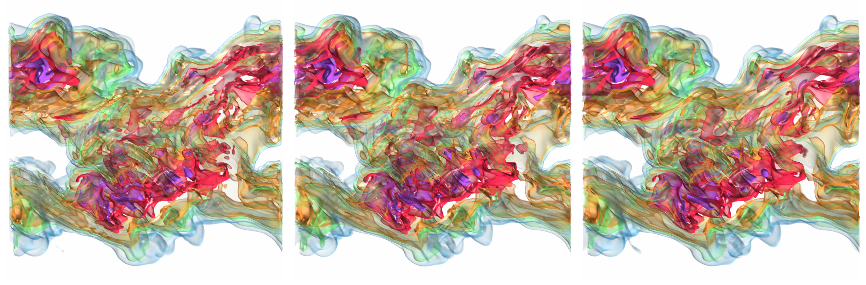

Fig. 7: Comparing different-variable inference results of the Combustion (MF $\rightarrow$ YOH) dataset using volume rendering. Left to right: bicubic, SU, and ground truth.

spatiotemporally coherent spatial super-resolution of the given volume sequence using adversarial learning. Formally, the goal is to estimate a mapping function $\mathcal{F}$ from a low-resolution volume sequence $\mathbf{V}^{LR}$ to a high-resolution volume sequence $\mathbf{V}^{EST}$, while taking into account temporal coherence. Namely, $\mathbf{V}^{EST} = \mathcal{F}(\mathbf{V}^{LR})$. The network is then trained by minimizing the loss function that considers (1) *adversarial loss* [35] which trains $G$ with the goal of fooling $D_s$ and $D_t$, (2) *content loss* [40, 23] which mixes the adversarial loss with a more traditional loss, such as $L_2$ distance, and (3) *feature loss* [59] which constrains $G$ to produce similar features between synthesized and ground-truth volumes at different scales.

**Preliminary Results.** For same-variable inference, Figure 5 compares the isosurfaces extracted from the synthesized volumes via bicubic interpolation and SU.

A single isovalue is picked and the results for a single timestep are shown. Clearly, SU generates closer results with respect to the ground truth. Figure 6 compares volume rendering of the synthesized volumes via bicubic interpolation, CNN, and SU. The CNN-based baseline model utilizes a post-upsampling framework [33]. It is clear that CNN generates the worst result while SU yields the best result. Besides same-variable inference, the framework can perform different-variable inference. That is, a variable $X$ of a dataset is used for training. For inference, $X$ is used to infer another variable $Y$ of the same dataset ($X \rightarrow Y$). Such an example is shown in Figure 7, where the MF variable is used for training and the YOH variable is used for inference. Again, SU produces higher-quality and more detailed visual results than bicubic interpolation.

## 5 Upscaling Scenarios - Temporal Upscaling

Temporal upscaling (TU) is particularly useful as a large-scale scientific simulation often runs a long sequence but could only afford to store the volume sequence rather sparsely (e.g., every 100th timestep). The upscaling aims to synthesize the intermediate timesteps by providing temporally resolved details to support a more detailed analysis and visualization of dynamic temporal features. A possible integration with in situ implementation is as follows. The simulation also first outputs, for example, 40% of samples from the entire volume sequence as the training data. The simulation then resumes and only sparsely outputs subsequent volumes (says every tenth or hundredth timestep), as the testing data. For network training, it takes pair-wise timesteps at two ends as input and aims to predict intermediate timesteps in between through forward and backward predictions (where intermediate timesteps are known for loss computation). Once training offline, the work can perform real-time inference to predict intermediate timesteps given a pair of timesteps in the testing set (where no intermediate timesteps are known).

The TU solution given by Han and Wang [18] uses a *recurrent generative network* that combines RNN and GAN, to generate the intermediate volumes between a given pair of volumes (i.e., two-end timesteps). Given a pair of volumes $(V_i, V_{i+k})$ from timesteps $i$ and $i + k$ (where $k > 1$), a function $\mathcal{F}$ is sought that satisfies $\mathcal{F}(V_i, V_{i+k}) \approx \mathbf{V}$, where $\mathbf{V} = \{V_{i+1}, V_{i+2}, \cdots, V_{i+k-1}\}$ are the intermediate volumes between $V_i$ and $V_{i+k}$.

**Network Design and Loss Function.** As sketched in Figure 8, the network includes a generator $G$ and a discriminator $D$. $G$ uses two modules: the *predicting module* ($\mathcal{F}_{PRED}$) and *blending module* ($\mathcal{F}_{BLD}$), to estimate $\mathcal{F}$. $\mathcal{F}_{PRED}$ is a volume prediction network that produces a forward prediction $\mathbf{V}^{FW}$ through $V_i$ and a backward prediction $\mathbf{V}^{BW}$ through $V_{i+k}$, respectively. $\mathcal{F}_{PRED}$ includes three components: (1) *feature learning component* which extracts feature representations from the volumes, (2) *temporal component*, which bridges the spatial and temporal information among different volumes, and *upscaling component*, which recovers the volumes from the spatiotemporal features. $\mathcal{F}_{BLD}$ takes $V_i$, $V_{i+k}$, and $V_{i+j}$ $(0 < j < k)$ from

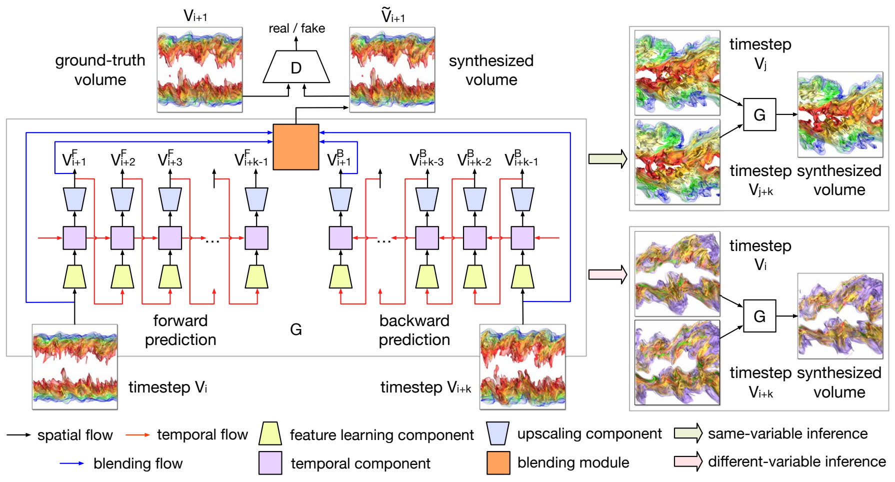

Fig. 8: The TU network has a generator $G$ (including the predicting and blending modules) and a discriminator $D$. During inference, the network performs either same-variable inference or different-variable inference using $G$. This image is adapted from our previous work [18].

$\mathbf{V}^{FW}$ and $\mathbf{V}^{BW}$ as input, and blends them into the predicted volume $\tilde{\mathbf{V}}$. The discriminator $D$ distinguishes $\tilde{\mathbf{V}}$ from $\mathbf{V}$. Given an input, $D$ produces a score to indicate whether or not the input is from the real data. That is, $D(\mathbf{V}) \approx 1$ and $D(\tilde{\mathbf{V}}) \approx 0$. Note that the goal of $G$ is to fool $D$ so that $D$ cannot distinguish $\tilde{\mathbf{V}}$ as fake volumes In this regard, $D$ serves the role of a binary classifier as the score from $D$ can guide $G$ in synthesizing high-quality volumes. Similar to spatial upscaling, the network is trained by minimizing the loss function that considers adversarial loss, content loss, and feature loss.

**Architecture Details.** With a traditional residual block, the resolution of the input [21] cannot be changed. Therefore, Han and Wang [18] opt to enhance the residual block by allowing downscaling or upscaling the input. For the feature learning component, each residual block is designed to contain two parts, which are bridged by skip connection. The first part has four convolutional layers, followed by spectral normalization [37] and ReLU [38]. The second part has one convolutional layer, followed by spectral normalization and ReLU. For the temporal component, to enable the network to predict volumes, ConvLSTM [46] is applied to transfer the spatial features into spatiotemporal features. The advantage of ConvLSTM over traditional LSTM [22] lies in its weight-sharing mechanism in convolution. Weight-sharing uses fewer parameters to train ConvLSTM, thus saving memory and speeding up training. For the upscaling component, the spatiotemporal features from ConvLSTM are taken as input and a synthesized volume is output. Although common, using deconvolutional layers to recover resolution from max-pooling or convolutional layers would incur a high computational cost. Therefore, right after

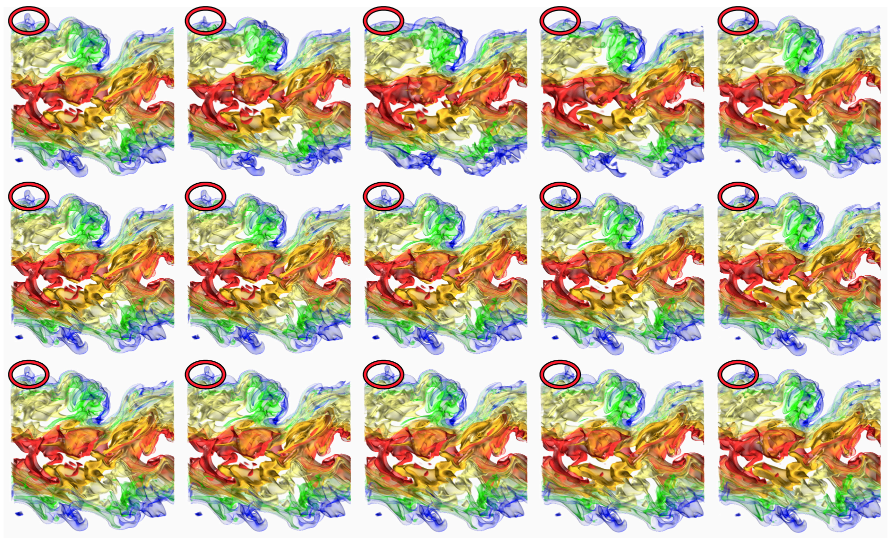
Fig. 9: Comparing same-variable inference results of the Combustion (MF) dataset using volume rendering. Top to bottom: linear, TU, and ground truth. Left to right: five consecutive timesteps. The synthesized results show that TU solution can preserve temporal coherence well while linear interpolation fails to do so (refer to the evolution of a volumetric feature highlighted in ellipses).

the last spectral normalization layer, Han and Wang [18] add the *voxel shuffle* layer, a sub-voxel convolutional layer, for upscaling. Assuming a feature of size $[L, W, H]$ needs to be upscaled with a factor $f$, voxel shuffle applies a periodic shuffle operation to rearrange the elements of a $[cf^3, L, W, H]$ tensor to a tensor of $[c, fL, fW, fH]$, where $c$ is the number of channels.

**Qualitative Results.** Figure 9 shows the comparison of volume rendering images of synthesized volumes generated using linear interpolation and inferred using TU. The ground-truth results are displayed for reference. The comparison shows that linear interpolation fails to capture the temporal evolution of features while TU can. For example, the ground-truth rendering image sequence shows that the volume feature highlighted in ellipses actually shifts from left to right (note that the feature at the two-end timesteps do not overlap spatially). However, due to the non-overlap of this feature at the two-end timesteps, linear interpolation would interpret this as two separate features: the feature on the left shrinks and disappears while the feature on the right appears and grows. TU can learn temporal evolution for accurate inference of intermediate timesteps. The different-variable inference results shown in Figure 10 also confirms the effectiveness of TU over linear interpolation.

Figure 11 compares the isosurfaces extracted from the synthesized volumes via linear interpolation and TU. A single isovalue is picked and the results for two different timesteps are shown. Compared with the ground-truth results, it can be seen that at timestep 33, linear interpolation generates a similar isosurface; at timestep

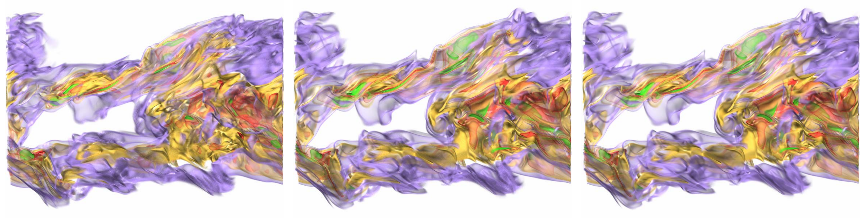

Fig. 10: Comparing different-variable inference results of the Combustion (MF → HR) dataset using volume rendering. Left to right: linear, TU, and ground truth.

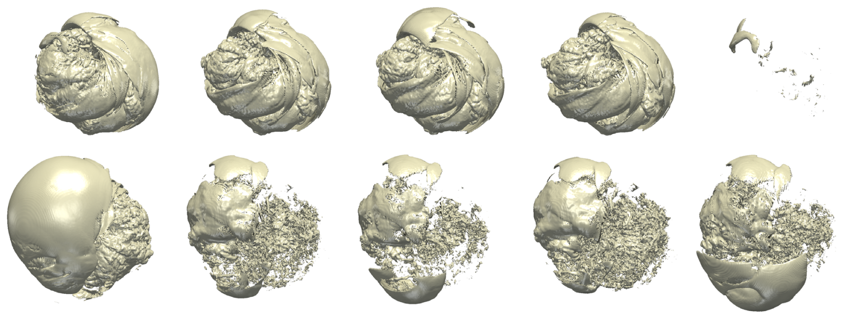

Fig. 11: Comparing same-variable inference results of the Supernova (E) dataset using isosurface rendering. The chosen isovalue is 0.255. Top row: timestep 35. Bottom row: timestep 51. Left to right: linear, TU, ground truth, and the two-end ground-truth timesteps (i.e., 33 and 37 for timestep 35, 49 and 53 for timestep 51).

51, linear interpolation fails to construct the isosurface. However, in both cases, TU clearly generates closer results with respect to the ground truth.

**Quantitative Results.** For quantitative comparison, PSNR and SSIM are used to evaluate the quality of synthesized volumes at the data and image levels, respectively. In addition to linear interpolation, two baseline deep learning solutions based on RNN and CNN are implemented. For the training of RNN, the architecture of TU is followed but the discriminator is excluded. For the training of CNN, the same TU architecture is leveraged but ConvLSTM is removed. Table 2 reports the average PSNR and SSIM values over the entire volume sequence for linear interpolation, RNN, CNN, and TU. At the data level, RNN performs the best in terms of PSNR. Such a result is expected, as RNN is a PSNR-oriented solution. In contrast, TU is also constrained by adversarial and feature losses. At the image level, TU performs the best in terms of SSIM.

Table 2: Comparing average PSNR and SSIM values for different methods.

| | PSNR (dB) | | | | SSIM | | | |
|---|---|---|---|---|---|---|---|---|
| dataset (variable) | linear | RNN | CNN | TU | linear | RNN | CNN | TU |
| Combustion (HR) | 25.61 | **26.13** | 25.72 | 25.81 | 0.66 | 0.70 | 0.69 | **0.72** |
| Combustion (MF) | 25.12 | **25.86** | 25.43 | 25.62 | 0.71 | 0.73 | 0.73 | **0.74** |
| Supernova (E) | 22.34 | **24.31** | 23.81 | 23.74 | 0.61 | 0.64 | 0.63 | **0.66** |
| Vortex | 26.62 | **27.42** | 26.85 | 26.90 | 0.73 | 0.75 | 0.75 | **0.75** |

## 6 Concluding Remarks and Future

The current results of deep learning-based data upscaling indicate that such methods have the potential to overcome some of the limitations in current high-performance simulation environments. The trained networks infer well the geometric properties of isosurfaces in 3D scalar fields, and they can even predict missing spatial and temporal features in static and time-resolved fields. As such, there is evidence that deep learning-based upscaling methods can (1) reduce the number of samples that need to be reconstructed (and transmitted in a remote environment) when monitoring a running simulation in situ, (2) reduce the resolution at which data needs to be stored, and (3) reduce the number of timesteps that need to be stored.

In computer vision, a single neural network model trained on various types of images could effectively upscale unseen images from multiple categories. However, this is not the case for scientific data since the training data is limited, and different scientific datasets may not follow a single data distribution (e.g., Gaussian distribution). Still, we believe that one can train a model on a certain type of datasets and later apply it to upscale or infer a different dataset of the same type (e.g., a different variable sequence or ensemble run). For example, Han et al. [20] recently designed V2V, a framework for variable to variable transfer using GAN.

For future work regarding the use of learning-based upscaling for in situ visualization, we envision in particular the following approaches.

When using 2D image-based upscaling in combination with 3D upscaling, it might be possible to let networks infer on the raw data from multiple (low-resolution) views of selected features in this data, facilitating a feature-based encoding of physical fields. An interesting research question is which features and how many of them are required by a network to infer the original dataset, and whether networks can locally infer the data from these features.

Even more important seems the question whether ANNs can convert the data to a compact feature-preserving representation (a code) that can be permanently stored, and directly decoded by the visualization tool into a visual representation, without having to decode the initial data. In this context, it will be interesting to revisit data compression techniques in light of the so-called “task-dependency principle" from psychology. This principle suggests that a code is optimal if it considers the behavioral goals of a user of this code, which is a perceptual investigation of the information encoded in the data when performing visual data analysis tasks. Following the task-dependency principle, codes should allocate resources according to how the user makes use of the encoded information, and the encoding of data that are irrelevant should be allocated minimal resources. This principle is fundamental to data visualization, since it asks for the reconstruction of data from a perceptual point of view, rather than a signal processing standpoint that argues in terms of numerical accuracy. It is an interesting question whether ANNs can generate such task-dependent, i.e., perception-aware, codes that can be intertwined with a visualization tool in the envisioned way. Answers to this question might be obtained by looking at recent works related to scene representation networks [36] and differentiable rendering [39], where reconstruction networks and networks that learn to generate rendered imagery from the resulting latent space representations have been trained end-to-end.

## References

1. Bai, K., Li, W., Desbrun, M., Liu, X. Dynamic upsampling of smoke through dictionary-based learning. arXiv preprint arXiv:1910.09166 (2019)
2. Bengio, Y. Learning deep architectures for AI. Foundations and Trends in Machine Learning, **2**(1), 1–127 (2009)
3. Caballero, J., Ledig, C., Aitken, A.P., Acosta, A., Totz, J., Wang, Z., Shi, W. Real-time video super-resolution with spatio-temporal networks and motion compensation. arXiv preprint arXiv:1611.05250 (2016)
4. Chaitanya, C.R.A., Kaplanyan, A.S., Schied, C., Salvi, M., Lefohn, A., Nowrouzezahrai, D., Aila, T. Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. ACM Transactions on Graphics, **36**(4), 98:1–98:12 (2017)
5. Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
6. Chu, M., Thuerey, N. Data-driven synthesis of smoke flows with CNN-based feature descriptors. ACM Transactions on Graphics, **36**(4), 69:1–69:14 (2017)
7. Chu, M., Xie, Y., Leal-Taixé, L., Thuerey, N. Temporally coherent gans for video super-resolution (TecoGAN). arXiv preprint arXiv:1811.09393 (2018)
8. Dong, C., Loy, C.C., He, K., Tang, X. Learning a deep convolutional network for image super-resolution. In: Proceedings of European Conference on Computer Vision, pp. 184–199 (2014)
9. Dong, C., Loy, C.C., He, K., Tang, X. Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, **38**(2) 295–307 (2016)
10. Dosovitskiy, A., Brox, T. Generating images with perceptual similarity metrics based on deep networks. In: Proceedings of Annual Conference on Neural Information Processing Systems, pp. 658–666 (2016)
11. Eckert, M.-L., Um, K., Thuerey, N. ScalarFlow: A large-scale volumetric data set of real-world scalar transport flows for computer animation and machine learning. ACM Transactions on Graphics, **38**(6), 239:1–239:16 (2019)
12. Gatys, L.A., Ecker, A.S., Bethge, M. Image style transfer using convolutional neural networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 2414–2423 (2016)
13. Goodfellow, I., Bengio, Y., Courville, A. Deep Learning. MIT Press (2016)

14. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. Generative adversarial nets. In: Proceedings of Annual Conference on Neural Information Processing Systems, pp. 2672–2680 (2014)
15. Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H., Schmidhuber, J. A novel connectionist system for improved unconstrained handwriting recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, **31**(5), 855–868 (2009)
16. Guo, L., Ye, S., Han, J., Zheng, H., Gao, H., Chen, D.Z., Wang, J.-X., Wang, C. SSR-VFD: Spatial super-resolution for vector field data analysis and visualization. In: Proceedings of IEEE Pacific Visualization Symposium, pp. 71–80 (2020)
17. Han, J., Tao, J., Zheng, H., Guo, H., Chen, D.Z., Wang, C. Flow field reduction via reconstructing vector data from 3D streamlines using deep learning. IEEE Computer Graphics and Applications, **39**(4), 54–67 (2019)
18. Han, J., Wang, C. 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. Han, J., Wang, C. SSR-TVD: Spatial super-resolution for time-varying data analysis and visualization. IEEE Transactions on Visualization and Computer Graphics, Under Minor Revision, (2020)
20. Han, J., Zheng, H., Xing, Y., Chen, D.Z., 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, **27**(2), In Press, (2021)
21. He, K., Zhang, X., Ren, S., Sun, J. Deep residual learning for image recognition. In: Proceedings of International Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
22. Hochreiter, S., Schmidhuber, J. Long short-term memory. Neural Computation, **9**(8), 1735–1780 (1997)
23. Isola, P., Zhu, J.-Y., Zhou, T., Efros, A.A. Image-to-image translation with conditional adversarial networks. In: Proceedings of International Conference on Computer Vision and Pattern Recognition, pp. 1125–1134 (2017)
24. Jo, Y., Oh, S.W., Kang, J., Kim, S.J. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 3224–3232 (2018)
25. Johnson, J., Alahi, A., Li, F.-F. Perceptual losses for real-time style transfer and super-resolution. In: Proceedings of European Conference on Computer Vision, pp. 694–711 (2016)
26. Kappeler, A., Yoo, S., Dai, Q., Katsaggelos, A.K. Video super-resolution with convolutional neural networks. IEEE Transactions on Computational Imaging, **2**(2), 109–122 (2016)
27. Kim, J., Lee, J.K., Lee, K.M. Deeply-recursive convolutional network for image super-resolution. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1637–1645 (2016)
28. Kim, J., Lee, J.K., Lee, K.M. Accurate image super-resolution using very deep convolutional networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1646–1654 (2016)
29. Kramer, M.A. Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal, **37**(2), 233–243 (1991)
30. Krizhevsky, A., Sutskever, I., Hinton, G.E. Imagenet classification with deep convolutional neural networks. In: Proceedings of Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
31. Lai, W.-S., Huang, J.-B., Ahuja, N., Yang, M.-H. Deep Laplacian pyramid networks for fast and accurate superresolution. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 624–632 (2017)
32. LeCun, Y., Bengio, Y., Hinton, G.E. Deep learning. Nature, **521**(7553), 436–444 (2015)
33. Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A.P., Tejani, A., Totz, J., Wang, Z., Shi, W. Photo-realistic single image super-resolution using a generative adversarial network. In: Proceedings of International Conference on Computer Vision and Pattern Recognition, pp. 4681–4690 (2017)

34. Liu, D., Wang, Z., Fan, Y., Liu, X., Wang, Z., Chang, S., Huang, T. Robust video super-resolution with learned temporal dynamics. In: Proceedings of International Conference on Computer Vision, pp. 2526–2534, (2017)
35. Mao, X., Li, Q., Xie, H., Lau, R.Y.K., Wang, Z., Smolley, S.P. Least squares generative adversarial networks. In: Proceedings of International Conference on Computer Vision, pp. 2813–2821 (2017)
36. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. arXiv preprint arXiv:2003.08934 (2020)
37. Miyato, T., Kataoka, T., Koyama, M., Yoshida, Y. Spectral normalization for generative adversarial networks. In: Proceedings of International Conference for Learning Representations, (2018)
38. Nair, V., Hinton, G.E. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of International Conference on Machine Learning, pp. 807–814 (2010)
39. Nimier-David, M., Vicini, D., Zeltner, T., Wenzel, J. Mitsuba 2: A retargetable forward and inverse renderer. ACM Transactions on Graphics, **38**(6), 203:1–203:17 (2019)
40. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A. Context encoders: Feature learning by inpainting. In: Proceedings of International Conference on Computer Vision and Pattern Recognition, pp. 2536–2544 (2016)
41. Ronneberger, O., Fischer, P., Brox, T. 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)
42. Rumelhart, D.E., Hinton, G.E., Williams, R.J. Learning representations by back-propagating errors. Nature, **323**, 533–536 (1986)
43. Sajjadi, M.S.M., Schólkopf, B., Hirsch, M. EnhanceNet: Single image super-resolution through automated texture synthesis. In: Proceedings of IEEE International Conference on Computer Vision, pp. 4501–4510 (2017)
44. Sajjadi, M.S.M., Vemulapalli, R., Brown, M. Frame-recurrent video super-resolution. In: Proceedings IEEE Conference on Computer Vision and Pattern Recognition, pp. 6626–6634 (2018)
45. Schmidhuber, J. Deep learning in neural networks: An overview. Neural Networks, **61**, 85–117 (2015)
46. Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., Woo, W.-C. Convolutional LSTM network: A machine learning approach for precipitation Nowcasting. In: Proceedings of Advances in Neural Information Processing Systems, pp. 802–810 (2015)
47. Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1874–1883 (2016)
48. Simonyan, K., Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
49. Tai, Y., Yang, J., Liu, X. Image super-resolution via deep recursive residual network. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 3147–3155 (2017)
50. Tao, X., Gao, H., Liao, R., Wang, J., Jia, J. Detail-revealing deep video super-resolution. In: Proceedings of IEEE International Conference on Computer Vision, pp. 4482–4490 (2017)
51. Tong, T., Li, G., Liu, X., Gao, Q. Image super-resolution using dense skip connections. In: Proceedings of IEEE International Conference on Computer Vision, pp. 4809–4817 (2017)
52. Wang, X., Chan, K.C., Yu, K., Dong, C., Loy, C.C. EDVR: Video restoration with enhanced deformable convolutional networks. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops, (2019)
53. Weiss, S., Chu, M., Thuerey, N., Westermann, R. Volumetric isosurface rendering with deep learning-based super-resolution. IEEE Transactions on Visualization and Computer Graphics, Accepted, (2019)

54. Weiss, S., Işık, M., Thies, J., Westermann, R. Learning adaptive sampling and reconstruction for volume visualization. arXiv preprint arXiv:2007.10093 (2020)
55. Werhahn, M., Xie, Y., Chu, M., Thuerey, N. A multi-pass GAN for fluid flow super-resolution. Proceedings of the ACM on Computer Graphics and Interactive Techniques, **2**(1), 10:1–10:21 (2019)
56. Xiao, X., Wang, H., Yang, X. A CNN-based flow correction method for fast preview. Computer Graphics Forum, **38**(2), 431–440 (2019)
57. Xie, Y., Franz, E., Chu, M., Thuerey, N. tempoGAN: A temporally coherent, volumetric GAN for super-resolution fluid flow. ACM Transactions on Graphics, **37**(4), 95:1–95:15 (2018)
58. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. arXiv preprint arXiv:1608.03981 (2016)
59. Zhang, R., Isola, P., Efros, A.A., 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, pp. 586–595 (2018)
60. Zhou, Z., Hou, Y., Wang, Q., Chen, G., Lu, J., Tao, Y., Lin, H. Volume upscaling with convolutional neural networks. In: Proceedings of Computer Graphics International, pp. 38:1–38:6 (2017)