Filling the Void: Data-Driven Machine Learning-based Reconstruction of Sampled Spatiotemporal Scientific Simulation Data

Ayan Biswas
Los Alamos National Lab
Los Alamos, U.S.A.
ayan@lanl.gov

Aditi Mishra
Arizona State University
Tempe, U.S.A.
amishr45@asu.edu

Meghanto Majumder
Los Alamos National Lab
Los Alamos, U.S.A.
meghanto@lanl.gov

Subhashis Hazarika
Fujitsu Research of America Inc.
Santa Clara, U.S.A.
shazarika@fujitsu.com

Alexander Most
Los Alamos National Lab
Los Alamos, U.S.A.
amost@lanl.gov

Juan Castorena
Los Alamos National Lab
Los Alamos, U.S.A.
jcastorena@lanl.gov

Christopher Bryan
Arizona State University
Tempe, U.S.A.
cbryan16@asu.edu

Patrick McCormick
Los Alamos National Lab
Los Alamos, U.S.A
pat@lanl.gov

James Ahrens
Los Alamos National Lab
Los Alamos, U.S.A
ahrens@lanl.gov

Earl Lawrence
Los Alamos National Lab
Los Alamos, U.S.A
earl@lanl.gov

Aric Hagberg
Los Alamos National Lab
Los Alamos, U.S.A.
hagberg@lanl.gov

***Abstract*—As high-performance computing systems continue to advance, the gap between computing performance and I/O capabilities is widening. This bottleneck limits the storage capabilities of increasingly large-scale simulations, which generate data at never-before-seen granularities while only being able to store a small subset of the raw data. Recently, strategies for data-driven sampling have been proposed to intelligently sample the data in a way that achieves high data reduction rates while preserving important regions or features with high fidelity. However, a thorough analysis of how such intelligent samples can be used for data reconstruction is lacking. We propose a data-driven machine learning approach based on training neural networks to reconstruct full-scale datasets based on a simulation's sampled output. Compared to current state-of-the-art reconstruction approaches such as Delaunay triangulation-based linear interpolation, we demonstrate that our machine learning-based reconstruction has several advantages, including reconstruction quality, time-to-reconstruct, and knowledge transfer to unseen timesteps and grid resolutions. We propose and evaluate strategies that balance the sampling rates with model training (pretraining and fine-tuning) and data reconstruction time to demonstrate how such machine learning approaches can be tailored for both speed and quality for the reconstruction of grid-based datasets.**

***Index Terms*—computing methodologies, modeling and simulation, simulation types and techniques, scientific visualization, computing methodologies, machine learning, machine learning approaches, neural networks.**

I. Introduction

As high performance computing (HPC) workflows begin to utilize exascale computing, the gap between the ability of such systems to produce massive amounts of data and existing networks to efficiently transport and store such data continues to widen. Scientific simulations can produce petabytes of data at each timestep with very high spatial and temporal resolution. Disk I/O is the bottleneck, as the time it takes to transport, store, and post-process the data is far outpacing the time it takes to produce it.

Data sub-sampling is a widely applied data reduction method that selects a subset of the dataset for storage. Several *in situ* sampling strategies that have been studied in the context of scientific simulations, including stratified random sampling [1], bitmap indexing [2], and adaptive sampling [3]. However, as HPC continues to include exascale machines, sampling strategies are becoming more aggressive—storing as little as 1% or even 0.1% of the data at each timestep. Recent sampling strategies [4], [5] are addressing this by weighting the importance given to data points when performing sampling. In this way, important data-driven features can be preserved when higher importance is assigned to the points associated with (or nearby to) potential features, which is important particularly for visualization tasks such as volume rendering and isosurface contouring, while allowing very low sampling rates.

The inverse of data sampling is *data reconstruction*: going from a sampled dataset back to the full-resolution dataset. For scientific simulations, a primary consideration of reconstruction is not only to obtain as similar a result to the original data as possible, but also to preserve important data features. Intrinsically, reconstruction algorithms imbue some amount of noise in the reconstructed data. A reconstruction algorithm is considered poor if the amount of noise added to the reconstruction is higher than the signal value present. As an example, linear interpolation based on Delaunay triangulation [5] is a

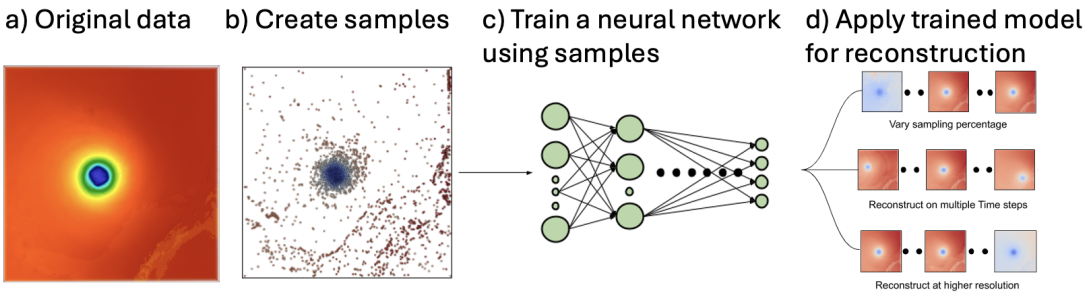


Fig. 1. The high-level workflow of our data-driven machine learning-based approach to reconstruct 3D volume datasets from sampled data. (a) Given an input regular grid dataset, (b) a sampled dataset is generated from it, and (c) a fully connected neural network is trained over the features extracted from the void locations. (d) The trained network is tested over varying sampling percentages, timesteps and at higher resolutions.

popular and well-regarded reconstruction algorithm with good signal-to-noise performance. However, it can suffer from two issues: (1) Its time complexity, which can increase rapidly with sampling percent, making it unsuitable for large-scale datasets such as those increasingly found in HPC scientific simulations. (2) It assumes a linear relationship within each cell while performing interpolation, which can result in sub-optimal results for regions with complex topology and high gradients.

Recently, machine learning (ML) and artificial intelligence (AI) approaches have been proposed for reconstruction [6], [7], employing complex deep learning architectures to predict a high-resolution data from a low resolution data. Thus far, such methods have been primarily applied to datasets that are highly structured and complete, i.e. where each data point has all the spatial and physical features present and thus can be easily defined. To illustrate this, lets assume that a dataset is sampled to store 1% of its data points, the remaining 99% of data points are discarded. In the sampling scenario, we lose the spatial features for even those 1% of data points since we now are dealing with missing data. The task of retrieving the missing 99% of data points (which we refer to as *void locations*) is non-trivial because the sampled data is now an unstructured point cloud. Performing convolution or using any machine learning algorithm which takes an image as input is not suitable. The result is that the existing AI/ML reconstruction approaches are incompatible with the aggressive sampling strategies being developed as part of exascale computing workflows.

This motivates the work described in this paper: *to develop and test the capabilities of a machine learning strategy to reconstruct from unstructured data, both with higher quality and for large datasets that are sampled with highly aggressive strategies*. Specifically, we are concerned with scientific simulations that are spatiotemporal in nature. These types of simulations are widespread, and due to their increasing resolution (as HPC systems advance and scale), they will soon necessitate aggressive sampling strategies that may only be able to preserve 0.1% or 1% of all raw data points at each timestep.

In this paper, we investigate fully connected neural networks (FCNNs) for reconstruction of unstructured sampled datasets that are used as input and structured reconstructed volume datasets are produced as output. For a comprehensive analysis of the reconstruction strategies, we employ popular point cloud-based reconstruction approaches that are based on Delaunay triangulation-based linear interpolation, nearest-neighbor, natural neighbors, radial basis function, Sheperd interpolation schemes, etc. (see Section III-B). After showing that Delaunay triangulation-based linear interpolation performs the best (as per reconstruction quality) amongst the existing approaches, we use it as a baseline for further comparison with our proposed FCNN approach. We test these approaches using three well-known scientific simulation benchmark datasets: Hurricane Isabel [8], combustion data [9], and the Ionization Front dataset [10] in a trio of experiments: (1) reconstruction based on varying the sampling percent for a single timestep, (2) training the FCNN on a single timestep and then reconstructing the dataset over multiple timesteps and (3) training on a low resolution dataset and reconstructing at higher resolutions.

Compared to Delaunay triangulation (or any other existing rule-based method), the FCNN approach has both advantages and disadvantages. The primary concern is initial training cost: as the size of a sampled dataset increases, the time required to train a model (pretraining) will also increase. However, the FCNN approach holds major advantages. A key finding in our experiments is that *a neural network trained on one timestep (pretrained network) can effectively reconstruct at different sampling percentages of that timestep*. This means that an FCNN can be trained once, and then reconstruct a simulation at different sampling percentages. We consistently see across the three studied datasets that our pretrained FCNNs maintain this key feature as well as overall high performance in comparison to Delauany triangulation, indicating that they provide a generalizable technique that can be adopted and replicated across scientific datasets.

Further, with very little *fine-tuning*, this pretrained model can be used in subsequent timesteps and also across different resolutions of a given simulation, still yielding much higher accuracy compared to the existing methods. Put another way, the cost of pretraining an FCNN can be amortized by usage on subsequent timesteps and resolutions. In contrast, rule-based methods like Delaunay triangulation must reconstruct from scratch at every timestep, yet does not produce the quality

that an FCNN can.

To summarize, the main contribution of our work are threefold:

- We develop a data-driven machine learning approach to convert unstructured sampled data to structured volume datasets with higher quality and rapid processing times compared to the current state-of-the art.
- We perform a comprehensive study of existing point cloud reconstruction strategies to explore the trade-offs between time complexity vs quality.
- We evaluate the robustness of our deep learning approach over different datasets, sampling percentages, timesteps, and reconstruction resolutions with minimal retraining of the neural network.

II. Related Work

Deep Learning in Scientific Visualization. In recent years there has been a steep increase in the use of various deep learning to address challenges in scientific visualization. Several studies include upscaling volume resolution by various sophisticated ML techniques. Zhou et al. [11] used a convolutional neural network (CNN) to achieve better quality upscaled resolution. The method has been proven to be better than traditional trilinear or cubic spline methods. Guo et al. proposed Ssr-vfd [6] which produces a spatially coherent high resolution data for vector field data by using three different neural networks. Following a similar line of work, Weiss et al. proposed [7] an image-space reconstruction of low-resolution images of isosurfaces to higher resolutions.

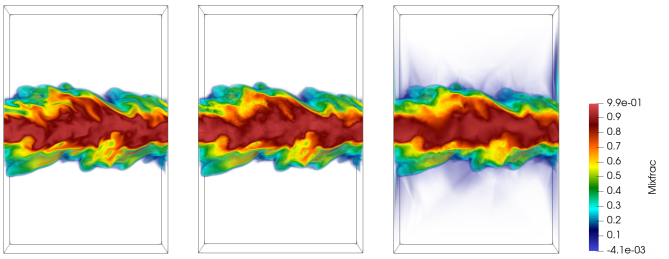
Another work which focuses on super resolution data generation is Han et al. [12], which uses a recurrent generative network (RGN) to generate high resolution volumes for temporal datasets from low resolution ones. It uses a generative network to produce volumes which then a discriminator decides the realness for. The network interpolates between two immediate volume sequences to give an output. Papers such as [13] and [14] focus on synthesizing volume by analyzing and playing around with the transfer functions. The former paper lets a user explore a latent space which encodes the effect of changing the transfer function on a volume rendering. This lets the user explore, analyze and generate volume data without an explicit mention of the transfer function. The latter paper avoids the exploration of transfer function so as to decrease cognitive load by training a deep neural network to obtain a goal effect image to obtain renderings under different viewing parameters without explicitly knowing the transfer function.

A similar paper in the lines of exploring the parameter space was proposed by He et al. [15] which is meant to generate volume data by exploring the parameter space for large ensemble simulations. Variations in simulation parameters under various visualization settings can help create new images. Hong et al. [16] used long short term memory (LSTM) based recurrent neural network (RNN) models to estimate the access patterns for parallel particle tracing in distributed environments. Using their trained model they predicted the next block to load while performing distributed particle tracing.

Several papers on flow field datasets have also been published which make use of several deep learning techniques. Han et al. [17] proposed an autoencoder framework to learn to cluster flow lines and surfaces. The autoencoder network learns the later feature descriptors from binary volumes generated from a flow field dataset. This paper also offers an interactive visualization tool to explore the flow lines and surfaces. Also an amalgamation for high resolution data for flow datasets was proposed by Xie et al [18] as TempoGAN which uses a temporal discriminator in addition to a spatial generator which preserves the data's temporal coherence. Weiwel et al. presented [19] an LSTM-based approach to predict dense volumetric time varying physical functions. However in all the mentioned methods, the data is structured and has no missing data points to deal with, thus enabling the usage of sophisticated machine learning techniques.

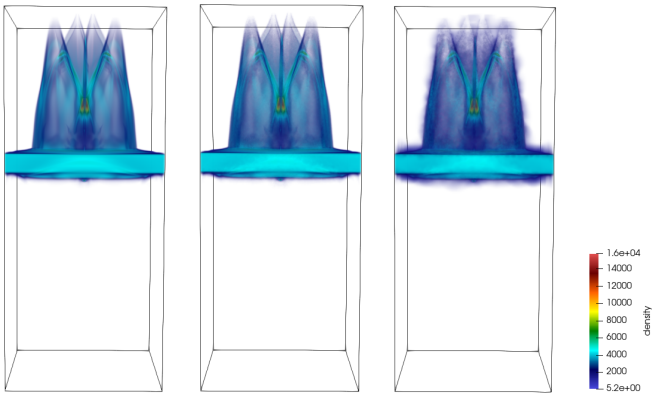
Sampling-based Visualization. Data sampling methods have been widely used in the scientific visualization community to reduce the size of large-scale data sets. Woodring et al. [1] proposed a stratified random sampling based algorithm for cosmology simulations to enable interactive post-hoc visualization. Su et al. [2] extended stratified sampling by incorporating value-based segmentation alongside spatial partitioning. Their method preserves both the value distribution and entropy of the original dataset, showing improved accuracy over random sampling and K-D tree based sampling for small bin sizes. Nouanesengsy et al. [3] developed Analysis-Driven Refinement (ADR), which uses user-defined importance metrics to select a sparse dataset for fast post-hoc analysis and visualization. Their approach employs either top-down or bottom-up subdivision schemes, considering space, field values, and time dimensions. Park et al. [20] proposed a visualization aware sampling technique which could produce an accurate complete visualization for scatter plot and map plot based visualizations. However this technique is specific for these visualization types, and does not generalize to 3D scientific simulation data.

In another work, Nguyen and Song [21] used a centrality-driven clustering approach to improve random sampling. Some other works which utilize information quantification techniques such as entropy have been proposed for sampling scientific datasets. Dutta et al. [22] proposed a point wise mutual information based approach for multivariate sampling to identify regions with high mutual information among the variables. Rapp et al. [23] proposed a method for scattered datasets which extracts a sample of points while preserving its blue noise properties. Biswas et al. [4], [5] proposed an in situ sampling technique which preserves important data features along with the gradient properties. This technique also ensures the extraction of important data features given a storage constraint. We tested this sampling method across various datasets and it showed good reconstruction quality when using Delaunay's method. We thus utilize the Biswas et al. [5] technique for all data sampling conducted in this work. Recent surveys, such as Di et al. [24], have highlighted sampling as one of the fastest approaches for data reduction



(a) Original data (b) Reconstruction via FCNN (c) Reconstruction via linear approach

Fig. 2. This figure shows how reconstruction quality for the combustion dataset (for 1% sampling percentage) differs between FCNN and Delaunay triangulation.



(a) Original data (b) Reconstruction via FCNN (c) Reconstruction via Natural Neighbors

Fig. 3. This figure shows how reconstruction quality for the Ionization Front dataset (for 1% sampling percentage) differs between FCNN and natural neighbors method.

in scientific visualization, particularly in the context of error-bounded lossy compression for scientific data.

While these existing approaches in machine learning and sampling-based visualization have made significant contributions, they are often limited when dealing with aggressively sampled, unstructured data from large-scale scientific simulations. This gap motivates our exploration of a novel FCNN-based approach that can effectively reconstruct full-resolution datasets from sparse, unstructured samples while maintaining high quality and efficiency across various sampling rates, timesteps, and resolutions.

III. Methodology

Figure 1 shows a high-level illustration of our proposed strategy. To learn the underlying features, we first extract a training dataset based on a sampled dataset and use this to train a fully connected neural network (FCNN). We test the capabilities of this trained model by experimenting with differing (1) sampling percentages, (2) timesteps for a single sampling percentage, and (3) data resolutions. For these experiments, we compare the reconstruction results using our FCNN model to the other available reconstruction strategies.

In this section, we first introduce the experimental datasets that we use for testing and evaluating the different reconstruction strategies. Next, we give a brief overview of FCNNs in the context of regression (predicting continuous values) and describe our specific FCNN implementation. We conclude this section with a brief overview of the feature engineering considerations for our FCNN approach (see Section V for more discussion on this).

A. Experimental Datasets

We employ three well-known scientific simulation datasets to test our neural network-based reconstruction. These datasets are representative of the types of data seen in scientific computing. For each dataset, we select an important scalar attribute to sample and reconstruct.

Hurricane Isabel: The Hurricane Isabel dataset [8] simulates the development of a hurricane in the West Atlantic region. It consists of eleven varying scalar and vector attributes. We test our method using the pressure attribute, which is indicative of the a hurricane's intensity [25]. This dataset has a resolution of $250 \times 250 \times 50$ over 48 timesteps. Figure 1 contains images of the Hurricane Isabel dataset. The eye of the hurricane—a very low pressure area—is the circular area in each sample image.

Combustion: The combustion dataset [9] simulates a turbulent combustion process. It consists of five scalar attributes. We test using the Mixfrac variable, which is the proportion of fuel and oxidizer mass. This attribute can indicates the flame's location where the chemical reaction rate exceeds the turbulent mixing rate. For our experiements, we use the combustion data that has a resolution of $240 \times 360 \times 60$ over 122 timesteps. Figure 2(a) shows an example of the combustion dataset.

Ionization Front Instabilities: The Ionization Front Instabilities dataset [10] simulates the propagation of an ionization front through neutral hydrogen gas. It consists of eleven varying scalar and vector attributes. The dataset has a resolution of $600 \times 248 \times 248$ over 200 timesteps. We use the density field for our experiments, as it shows the structure and evolution of the ionization front over time. The density values range from very low in the ionized regions to higher values in the neutral gas and compressed shell ahead of the front.

B. Reconstruction of Sampled Point Cloud

Several methods exist for reconstructing sampled point cloud data into a continuous field. We briefly describe some of the most common approaches:

- **Piecewise Linear Interpolation:** This method typically uses Delaunay triangulation of the sampled points and performs linear interpolation within each triangle or tetrahedron. It may result in artifacts at the triangle boundaries.
- **Natural Neighbor Interpolation:** Also known as Sibson's method, this approach uses the concept of natural neighbors based on Voronoi diagrams. As described by Park et al. [26], traditional implementations are computationally expensive, but discrete approaches have been

developed to improve efficiency while maintaining the method's smooth interpolations.

- **Modified Shepard Interpolation:** This inverse distance weighting method, an improvement over the original Shepard's method, uses only a subset of neighboring points and modifies the weighting function. It provides a balance between smoothness and local feature preservation, and is implemented in python packages such as photutils [27].
- **Nearest Neighbors:** This method assigns to each grid point the value of the nearest sampled point. While computationally efficient, it can result in discontinuities and a blocky appearance, especially with sparse sampling.
- **Radial Basis Functions (RBFs):** RBFs, such as thin-plate splines, can provide smooth interpolations [28]. However, they are often computationally impractical for large datasets and may produce poor results in some cases, especially when dealing with scattered data or extrapolation. For our experiments, we will not consider RBFs as the time taken by them is much larger than the rest of the methods, and it does not offer any noticeable improvement in reconstruction quality over linear interpolation.

C. An Overview of Our Fully Connected Neural Network

Fully connected neural networks, or FCNNs, are a class of artificial neural networks where the architecture is such that all the nodes (or neurons) in each layer L_i are connected to the nodes (neurons) in the next layer L_{i+1}. An FCNN with n layers has three types of layers: (1) An *input layer* (L_1) whose values are provided, normally described as an input feature vector. (2) A set of intermediate *hidden layers* ($L_2 - L_{n-1}$) whose values are derived from previous layers. (3) An *output layer* (L_n) whose values are derived from the last hidden layer. In our case, we are interested in predicting continuous numerical values (regression).

Iteratively calculating the values of neurons at each layer till we reach the output layer is called *forward propagation*. In our case, we used ReLU activation. To train an FCNN, the final outputs of a model for data items computed via forward propagation are compared to a ground truth based on an error or loss function. We use mean square error (MSE) to calculate the loss.

$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

By minimizing the calculated error, the weights between neurons are optimized via a process called *back propagation*—this is the "learning" part of the neural network. Back propagation computes the error gradients of each neuron, iteratively for each layer from the output layer backwards. We use the learning rate of 0.001 and the $adam$ optimizer for our network.

D. Training Dataset and FCNN Architecture

Now we describe the feature engineering part of the FCNN, which is a non-trivial task for predicting the missing values. During a large-scale scientific simulation, it is reasonable that the full-resolution dataset is only available for the current timestep (this is common for *in situ* workflows). Therefore, training data should be confined to a single available timestep. Figure 4 shows the workflow we use to train an FCNN for scientific data reconstruction.

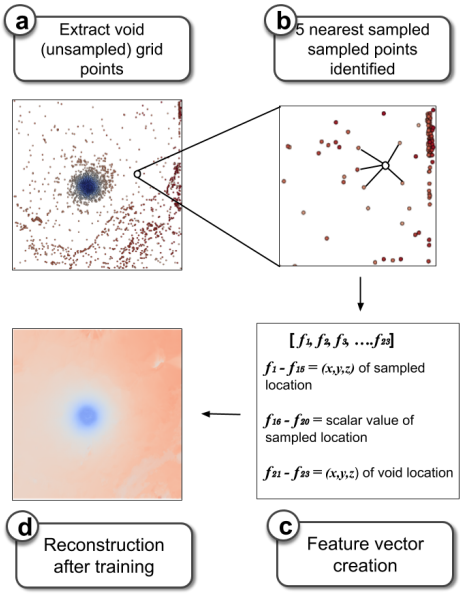


Fig. 4. The process to create a training dataset is as follows: (a) Extract void location values, and (b) for each, find the five nearest sampled points. (c) The *x, y* and *z* coordinates of each of the five nearest sampled points, the coordinates of the corresponding void point and the scalar values associated with those five sampled points is concatenated to form a $[1 \times 23]$ input feature vector. (d) The training data created is then sent as an input to the FCNN to give a reconstructed data as an output.

The dataset's grid points for the timestep are divided into two groups, *sampled points* and *void locations*, based on whether or not each grid point is included in the sampling set. By void locations, we refer to grid points in the dataset which were rejected by the sampling algorithm. Such void or empty grid locations lack the scalar value which acts like missing data and essentially the task of reconstruction is to predict these missing values. As discussed in Sec IV-A, for our experiments we use the state-of-the-art sampling method from Biswas et al. [5], though our approach is sampling method agnostic.

For feature engineering, we adopt an approach that is inspired by the k-nearest neighbor algorithm. For each void location from the set of rejected points, the five nearest sampled points are identified and a $[1 \times 23]$ feature vector is created consisting of (1) the x, y, and z coordinates and scalar values for each of the five closest sampled points (feature size = $5 * 4 = 20$) and (2) the x, y, and z coordinates of the void location itself (feature size = 3). The FCNN's input layer therefore consists of 23 $(= 20 + 3)$ neurons corresponding to the $[1 \times 23]$ vector created for each void point. The output layer predicts a $[1 \times 4]$ vector: the scalar value and the x, y, and z gradients of the void locations. The combined set of input and output vectors for all void locations constitutes the training dataset for the FCNN. For the FCNN architecture (Figure 5),

five hidden layers are used.

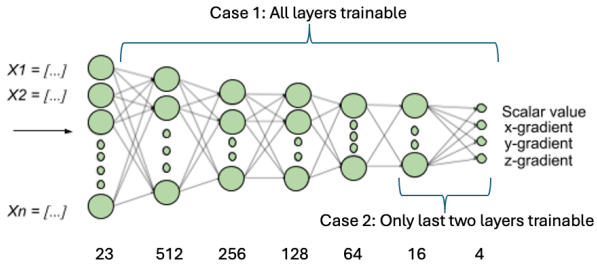


Fig. 5. The FCNN architecture that we utilize is composed of five hidden layers of size 512–16, and outputs a scalar value as well as the x-, y-, and z-gradients for a *void location*. This figure also shows the two approaches of performing fine-tuning from a pre-trained model. Case 1: we set all the layers as trainable and this enables us to perform fewer epochs (10 epochs) to fine-tune to the new unseen data. Case 2: we set only the last two layers trainable that enables lower storage across new timesteps (if we want to carry around models for all the time steps) but requires more epochs (≈300-500 epochs) of fine-tuning to achieve similar accuracy as Case 1.

E. Selecting and Tuning the FCNN

The previous subsection describes the "final" FCNN architecture (also shown in Figure 5) we employ for data reconstruction in this paper. Here, we briefly address why FCNNs are selected as an architecture, the feature engineering employed for the training dataset (i.e., creating the $[1 \times 23]$ vector using the five sampled points closest to each unsampled grid point), and the architectural considerations for the FCNN.

Why FCNNs? FCNNs are sometimes considered as "straightforward" or "simple" neural network architectures, as each layer is fully connected, they do not contain specialized layers (convolutional layers, pooling layers, etc.), and neurons do not incorporate any notion of temporal sequence. Despite this, FCNNs provide an appropriate model space for exploring reconstruction, in contrast to more complex models such as CNNs and RNNs. This is due to two key points: (1) The aggressive sampling strategies employed for scientific computing result in unstructured point-based datasets with a large number of void or missing data points. CNNs, which are widely used for images, generally require complete and structured data for pooling and convolution functions. (2) For large-scale simulations running on distributed HPC resources, it is impractical to store multiple timesteps for training models. Deep learning models that are trained in situ cannot take advantage of dataset temporality, which is the key feature of RNNs.

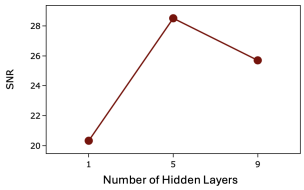


Fig. 6. Average SNR values when varying the number of hidden layers on the Hurricane Isabel dataset.

Choosing an appropriate number of hidden layers. Model complexity can have a large effect on the overall performance of the neural network. For example, a very simplistic model might underfit the data, resulting in high training error. In contrast, an overly complex model might overfit, resulting in a very low train error but an extremely high test error (ultimately making the model not generalizable). Along these lines, a large number of hidden layers has the potential to not just capture the data feature dynamics, but also to capture spurious statistical noises or biases in the data [29]

To find the appropriate number of hidden layers, we tested reconstructions with different numbers of hidden layers (between one and nine hidden layers). Figure 6 shows the results for the Hurricane Isabel dataset. Reconstruction quality (measured as signal-to-noise ratio, or SNR, see Section IV) is plotted against the number of hidden layers in the FCNN. The SNR for both one hidden layer (SNR$\approx$ 20) and nine hidden layers (SNR$\approx$ 25) is lower than for five layers (SNR$\approx$ 28). Our assumption is that an FCNN with one hidden layer does not learn the features very well because of a high bias (too simple of a model). In contrast, an FCNN with nine layers likely overfits the training data. In addition, a larger model increases the FCNN's training time. Five hidden layers achieves high quality while minimizing training time and the potential to overfit.

Sampling points. *Sampling percentage* refers to the total percent of data points that are saved in relation to the dataset. As the sampling percentage increases, sampled points will generally be closer to each other, while decreasing sampling percentage will result in sampled points being farther apart. Our assumption is that an FCNN will, when weights are assigned to features, employ higher weights to sampled points that are closest to the void location, compared to sampled points which are farther away. We want the FCNN to provide good reconstruction results for these void locations over a range of sampling percentages, which might range from sub-1% to 5% or 10% of the original dataset.

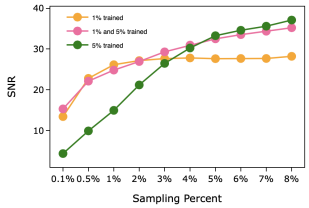


Fig. 7. Average SNR values when training on different sampling percentages on the Hurricane Isabel dataset.

Figure 7 shows how varying the sampling percentage during training affects reconstruction quality (SNR) when performing reconstruction for different sampling percents on the Hurricane Isabel dataset. When the model is trained on a 1% sampling of the data (orange line) and used to reconstruct a dataset that has been sampled at 1% or lower, SNR values are high, but reconstruction quality flatlines as the sampling percentage increases to 3% and higher. The reason is that as sampling percent increases, the distance between sampled points decreases. This behavior is not captured well using a 1% trained model, which assumes sampled points are spaced far apart. The opposite effect occurs when the model is trained on 5% of data (green line). Data reconstruction using higher sampling percentages (4% and higher) have high SNR, but at lower sampling percentage the model fails to capture the large distances between sampled points. As a solution, we

tested combining data points from both the 1% and 5% sampling percentages into a concatenated dataset (pink line). This "1%+5% model" provided good results at both ends of the sampling spectrum, and is what is used in our FCNN design.

Gradients in Output Layer: Incorporating gradient values along with the scalar values in the output layer helps achieve better reconstruction quality as compared to when utilizing only the scalar values. By including gradient values in the output layer, the FCNN model is forced to take into consideration the neighbouring locations' values while predicting the scalar value at a given location. For example, two different spatial locations can have the same scalar values, but depending on the distribution of the data features, they may have different gradient values. Our FCNN approach takes this factor into account when reconstructing the full scalar fields from sampled points. Figure 8 shows the result of an experiment we performed to empirically verify the influence of producing gradient values in the output layer. We performed this study by removing the gradient neurons from the output layer to compare the reconstruction quality with our proposed FCNN. The pink and the green curves correspond to the SNR values of reconstruction with increasing sampling percentage for "with gradient" and "without gradient" networks respectively. As can be seen, having the gradients in the output layer improves the overall reconstruction quality of our FCNN model.

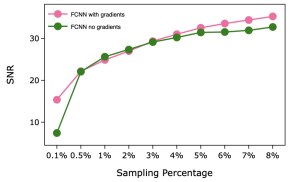


Fig. 8. SNR values for gradient vs no gradient scenarios in the output layer of the FCNN.

IV. Experiments and Results

We test our FCNN approach against existing point cloud reconstruction approaches to reconstruct from sampled data back to a full-resolution dataset. To calculate the reconstruction quality, we use the popular signal-to-noise ratio (SNR) defined as follows:

$$SNR = 20 * \log_{10} \frac{\sigma_{raw}}{\sigma_{noise}}$$

Here, σ_{raw} is the standard deviation of the original data and σ_{noise} is the standard deviation of the noise in the reconstructed data. Noise in a dataset is the difference among the values of the original data and the reconstructed data. A good reconstruction is represented by having a lower noise, which means the reconstructed image is similar to the original data. Thus, a well reconstructed image has higher SNR.

In general, for all the methods, a higher sampling percentage will lead to a better reconstruction. To investigate this, we perform three different experiments. We are interested not only in seeing if FCNN has a better reconstruction quality than the others, but want to see the time required by our FCNN approach to reconstruct the dataset.

A. Testing Setup

We conducted our experiments using the Darwin HPC cluster at Los Alamos National Laboratory. This compute cluster contains approximately 350 heterogeneous compute nodes. For our experiments, we used a single compute node containing 64 CPU cores and two NVIDIA A100 GPUs. When conducting experiments, we first use the sampling technique by Biswas et al. [5] to sample the data for a given sampling percentage (between 0.1% and 5%). This converts the original regular grid dataset, stored as a .vti file (XML Image data), into a point cloud dataset stored using the .vtp file format (XML Poly data). All the reconstruction methods are applied on this sampled dataset, and output the reconstructed dataset as .vti format (regular grid). To calculate SNR, we compare scalar values in the reconstructred .vti file to the true scalar values in the original .vti file(s).

B. Experiment 1: Varying Sampling Percentages

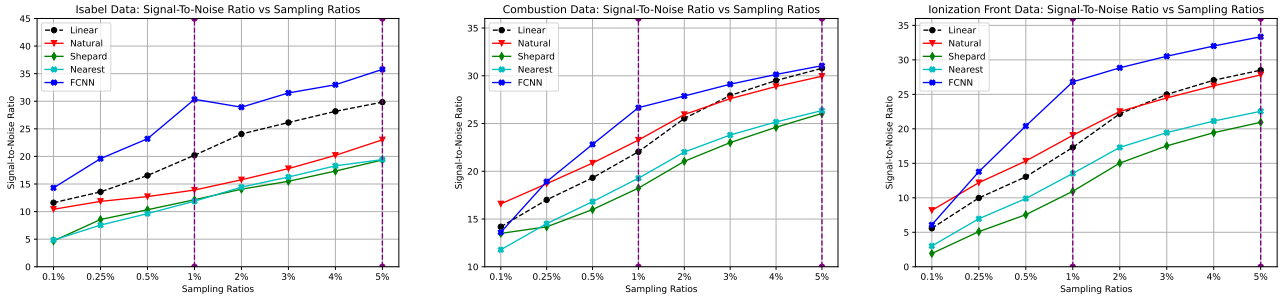
The first experiment compares reconstruction quality over different sampling percentages (also called sampling ratios) at a single timestep. We trained the FCNN model as explained in Section III. We then tested reconstruction on the three datasets with sampling percentages from 0.1% to 5%. Figure 9 shows the reconstruction quality and Figure 10 shows the corresponding time to reconstruction.

For all the methods, reconstruction quality generally increases as the sampling percentage increases. At 0.1% sampling percentages, FCNN, linear and natural neighbor approaches have comparable SNR. However, SNR is generally higher for FCNN. Natural neighbor and linear approaches are comparable at lower sampling percentages, and as we increase the sampling percentage, linear generally outperforms natural neighbor approach. Shepard and nearest neighbors approaches consistently yield lower quality.

In analyzing reconstruction time, we first note that Figure 10 does not include model training time—this is considered in Section V. For a trained FCNN, reconstruction occurs in constant time (with respect to the full dataset size) and is independent of the sampling percent. This highlights a key benefit to this experiment: the flexibility of FCNNs. For each dataset, we use a single FCNN to reconstruct across all sampling percentages. For all datasets, we see that FCNN's speed is comparable with nearest neighbor approach and parallelized version of linear interpolation using C++, CGAL and OpenMP. While linear interpolation provided one of the best SNR values among the traditional reconstruction methods, its initial sequential implementation in Python was much slower than other methods. The parallelized version was implemented to address this, leading to speedups that scaled with the number of processing units.

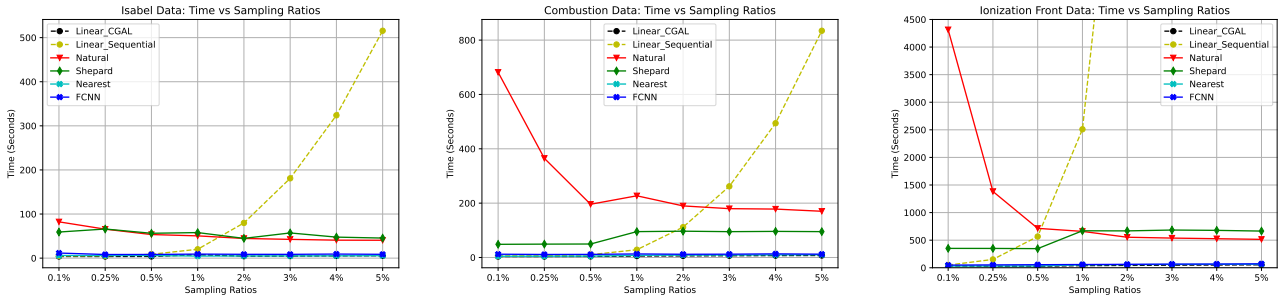
C. Experiment 2: Testing Over Multiple Timesteps

Next we show how an FCNN pretrained on one timestep can be used to reconstruct on other timesteps with and without fine-tuning. We focus on the Hurricane Isabel dataset (using a



(a) Hurricane Isabel (t=25) (b) Combustion ($t = 41$) (c) Ionization Front ($t = 99$)

Fig. 9. Reconstruction quality (SNR) for FCNN, linear, natural neighbor, Shepard and nearest neighbor approaches at different sampling percentages for a single timestep t. The two purple lines show the sampling percentages (1% and 5%) used by the FCNN approach while making predictions.



(a) Hurricane Isabel (t=25) (b) Combustion ($t = 41$) (c) Ionization Front ($t = 99$)

Fig. 10. Reconstruction time in seconds for FCNN, linear (both naive sequential and OpenMP CGAL implementations), natural neighbor, Shepard and nearest neighbor approaches at different sampling percentages for a single timestep t. Since naive linear method implementations scale poorly with data size, our CGAL implementation is necessary for handling large-scale data.

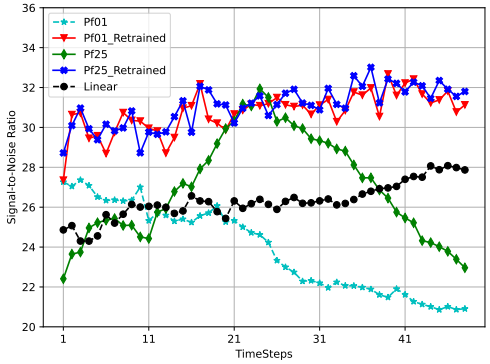
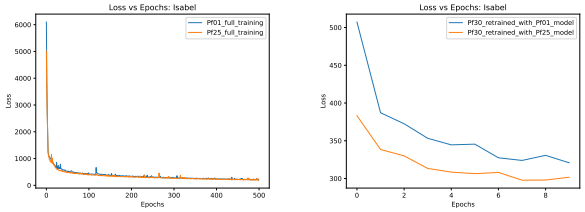


Fig. 11. This figure shows reconstruction quality for the Hurricane Isabel dataset over its 48 timesteps using a 3% sampling percentage at each timestep.



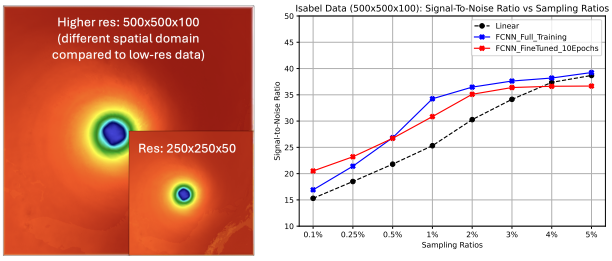
(a) Full training (b) Fine-tuning

Fig. 12. Loss progression across epochs for a) full training and b) fine-tuning.

3% sampling percentage) as this simulation showcases a complex weather pattern with features that change significantly over the course of the run as the hurricane moves across the Gulf of Mexico and makes landfall.

For fine-tuning, we adopt two approaches: Case 1 is full layer retraining to new time steps, and Case 2 is transfer learning via two layer retraining (this is shown in Figure 5). For Case 1, we only perform retraining for a few epochs (≈10 epochs) to fine-tune to the new time step. For Case 2, to reach the accuracy of Case 1, we need to perform more than 300 epochs. Compared to how fast we can fine-tune to the new time step, there is a trade-off involved in the size of the model. For Case 1, if we want to store all the models across all the time steps, we will need to carry each time step's individual fine-tuned model. For Case 2, since only the last two layers change from one time step to the other, only the first timestep needs to store the full model and for the remaining time steps, we can just store the last two layers. Further, for Case 1, since we are able perform very fast fine-tuning of around 10 epochs, there is generally no need to store individual models. We can store one model and then quickly fine-tune it as needed with newer data from other timesteps.

In Figure 11, we compare the reconstruction quality across 48 time steps of Isabel for 3% sampling percentages. We use linear approach as a baseline (shown in black), and use two FCNN models that are trained on time steps 01



(a) Hurricane Isabel dataset (b) SNR vs sampling percentages for low-res to high-res fine-tuning.

Fig. 13. Applying fine-tuned model from lower resolution Isabel data to its higher resolution data.

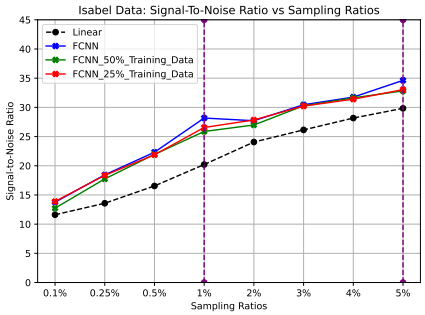


Fig. 14. This figure shows reconstruction quality for the Hurricane Isabel dataset when varying training samples are used during full training. The effect of using fewer samples during training is very small on the quality, but as per Table II, the training can be reduced almost linearly.

($FCNN_{Pf01}$, shown in cyan) and 25 ($FCNN_{Pf25}$, shown in green) and use them to predict on all 48 time steps. We see that $FCNN_{Pf01}$ performs well in the beginning timesteps, but continues to degrade in performance as the data changes over time. Similarly, $FCNN_{Pf25}$ performs the best around timestep 25 and degrades both going forward and backward. When we perform 10 epochs of retraining using these two models (red and blue) over the timesteps, we observe that these models fine-tune very quickly to the new data, and perform much better than linear. This shows the efficacy of the FCNN models over traditional rule-based approaches such as linear interpolation. The loss progressions during the full training and fine-tuning epochs are shown in Figure 12.

D. Experiment 3: Reconstruction Volume Upscaling

Generation of higher resolution data from lower resolution is called *volume upscaling*. This problem has been extensively studied in the field of scientific visualization, as discussed in Section II. For the third experiment, we wanted to test volume upscaling: Could we train an FCNN on a lower resolution dataset and apply it to reconstruct the samples taken from a higher resolution data? For this we used the Hurricane Isabel dataset. As mentioned in Section III-A, the dataset has a resolution of $250 \times 250 \times 50$. We intended to replicate Experiment 1 with this dataset, but now reconstructing to the resolution size: $500 \times 500 \times 100$. Thus we can test reconstruction quality by computing SNR against a resolution that is $2\times$ upscaled on each dimension (and hence a dataset that is $8\times$ larger). Further, we modified the spatial extent of the higher resolution data such that the higher resolution data spans across different domains as compared to the lower resolution data (Figure 13a). This experiment intended to not only test the upscaling capabilities, but also to test if different physical domains can be learnt by the fine-tuned FCNN model.

The SNR results are presented in Figure 13b. Again, the black curve represents the quality of the linear method. The blue curve shows the FCNN that is fully trained on higher resolution data. The red curve shows the results of the model that was created from lower resolution data and then fine-tuned for 10 epochs. This result further shows how the knowledge from lower resolution can be "transferred" to higher resolution and across different spatial domain.

V. Discussion and Conclusion

The current work shows that FCNNs are a viable solution for dataset reconstruction. While the results are very promising, we consider that there are ample opportunities to investigate how machine learning can be used and refined for reconstruction of scientific simulation datasets. We discuss the following points:

TABLE I
Training time for 500 epochs for the different datasets and resolutions.

Dataset	Resolution	Training Time (seconds)
Isabel	250x250x50	533
Isabel	500x500x100	3737
Combustion	240x360x60	829
Ionization Front	600x248x248	5522

TABLE II
Effect of sampling on training time for 500 epochs for Isabel dataset.

Dataset	% of Full Training Data	Training Time (seconds)
Isabel	100	533
Isabel	50	275
Isabel	25	161

Training data point selection The time to train a neural network is highly dependent both on the computer hardware to be used and on the number of training samples that are available. If all the training data were used (1%+5% samples), then the training time is listed in Table I. We experimented with reducing this training time further by random sampling the training set used to build the FCNN. The timing results are shown in Table II and the corresponding SNR results are shown in Figure 14. The results indicate that the decrease in quality (compared to use the full training dataset) was negligible, and even when using 50% and 25% of the samples for training. As a future work, we plan to formally investigate the idea of intelligent training set creation to support better and faster reconstruction.

Contributions, Limitations and Future Work. The experiments illustrate key benefits of using FCNNs to reconstruct scientific datasets: (1) reconstruction quality is generally better, (2) once trained, a model reconstructs extremely fast and in constant time, and (3) models can be saved and very quickly

fine-tuned to use for reconstruction at different timesteps, sampling percentages, and even dataset resolutions spanning different spatial domains. In addition to these experimental findings, we contribute the process of designing and fine-tuning a well thought-out FCNN architecture (see Section III).

Despite these advantages, we also see limitations and challenges with the current approach. (1) The first is training time. As shown previously, full retraining can be a little time consuming, but this gets amortised over the usages. Also, subsampling the training data can improve the full training time without lowering the prediction quality too much. (2) A second drawback is dataset specificity, as the FCNN is trained on the dataset it reconstructs. As a future work, we intend to explore how machine learning models can be trained to generalizably reconstruct varied simulation datasets. (3) A third challenge in our approach (also shared by other reconstruction methods) is uncertainty. One solution is investigating neural networks that include measures of uncertainty during reconstruction (e.g., using deep ensembles, Bayesian neural networks etc.). We additionally plan to explore this as a part of our future work.

VI. Conclusion

At current, the use of machine learning for reconstructing point-based scientific datasets is underexplored. To our knowledge, this is the first work to delve into this specific topic. Despite some acknowledged limitations (e.g., model training time), our results are especially encouraging in two ways: (1) Even with "straightforward" FCNN architectures, we can create an effective data-driven reconstruction approaches that are in many ways better than the state-of-the-art methods. (2) The current work establishes some key benefits of deep learning approaches (e.g., we can train and reconstruct at different scales), providing a foundation by which future work will likely greatly improve upon these results by creating much more sophisticated, flexible, and generalizable models. Finally, this work is well-timed, particularly as the age of exascale computing necessitates inventive ways to store, transfer, reconstruct, and analyze the massive amounts of data that will be generated by scientific simulations. As deep learning increasingly becomes a part of HPC workflows, experiments such as the ones presented here will be critical to understanding how and when to apply such techniques.

Acknowledgment

This research was supported by the Laboratory Directed Research and Development program of Los Alamos National Laboratory under project numbers 20230771DI and 20250637DI (LA-UR-24-29131).

References

[1] J. Woodring, J. Ahrens *et al.*, "In-situ sampling of a large-scale particle simulation for interactive visualization and analysis," in *Proceedings of the 13th Eurographics IEEE-VGTC Conference on Visualization*. Eurographics Association, 2011, pp. 1151–1160.

[2] Y. Su, G. Agrawal *et al.*, "Taming massive distributed datasets: data sampling using bitmap indices," in *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, 2013, pp. 13–24.

[3] B. Nouanesengsy, J. Woodring *et al.*, "Adr visualization: A generalized framework for ranking large-scale scientific data using analysis-driven refinement," in *LDAV 2014*. IEEE, 2014, pp. 43–50.

[4] A. Biswas, S. Dutta *et al.*, "In situ data-driven adaptive sampling for large-scale simulation data summarization," in *ISAV 2018*. New York, NY, USA: Association for Computing Machinery, 2018, p. 13–18.

[5] A. Biswas, S. Dutta, E. Lawrence *et al.*, "Probabilistic data-driven sampling via multi-criteria importance analysis," *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[6] L. Guo, S. Ye, J. Han, H. Zheng, H. Gao, D. Z. Chen, J. Wang, and C. Wang, "Ssr-vfd: Spatial super-resolution for vector field data analysis and visualization," in *2020 IEEE PacificVis*, 2020, pp. 71–80.

[7] S. Weiss, M. Chu, N. Thuerey, and R. Westermann, "Volumetric isosurface rendering with deep learning-based super-resolution," *IEEE TVCG*, pp. 1–1, 2019.

[8] http://vis.computer.org/vis2004contest/data.html.

[9] http://vis.cs.ucdavis.edu/Ultravis11/.

[10] D. J. Whalen and M. Norman, "Ionization front instabilities in primordial hiiregions," *Astrophysical Journal*, vol. 673, 2008.

[11] Z. Zhou, Y. Hou, Q. Wang, G. Chen, J. Lu, Y. Tao, and H. Lin, "Volume upscaling with convolutional neural networks," in *Proceedings of the Computer Graphics International Conference*, 2017, pp. 1–6.

[12] J. Han and C. Wang, "Tsr-tvd: Temporal super-resolution for time-varying data analysis and visualization," *IEEE TVCG*, vol. 26, no. 1, pp. 205–215, 2020.

[13] M. Berger, J. Li, and J. A. Levine, "A generative model for volume rendering," *IEEE TVCG*, vol. 25, no. 4, pp. 1636–1650, 2019.

[14] F. Hong, C. Liu, and X. Yuan, "Dnn-volvis: Interactive volume visualization supported by deep neural network," in *2019 IEEE Pacific Visualization Symposium (PacificVis)*, 2019, pp. 282–291.

[15] W. He, J. Wang, H. Guo, K. Wang, H. Shen, M. Raj, Y. S. G. Nashed, and T. Peterka, "Insitunet: Deep image synthesis for parameter space exploration of ensemble simulations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 23–33, 2020.

[16] F. Hong, J. Zhang, and X. Yuan, "Access pattern learning with long short-term memory for parallel particle tracing," in *2018 IEEE Pacific Visualization Symposium (PacificVis)*, 2018, pp. 76–85.

[17] J. Han, J. Tao, and C. Wang, "Flownet: A deep learning framework for clustering and selection of streamlines and stream surfaces," *IEEE TVCG*, vol. 26, no. 4, pp. 1732–1744, 2020.

[18] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "Tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018. [Online]. Available: https://doi.org/10.1145/3197517.3201304

[19] S. Wiewel, M. Becher, and N. Thuerey, "Latent space physics: Towards learning the temporal evolution of fluid flow," *Computer Graphics Forum*, vol. 38, no. 2, pp. 71–82, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13620

[20] Y. Park, M. Cafarella, and B. Mozafari, "Visualization-aware sampling for very large databases," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 755–766.

[21] T. T. Nguyen and I. Song, "Centrality clustering-based sampling for big data visualization," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 1911–1917.

[22] S. Dutta, A. Biswas, and J. Ahrens, "Multivariate pointwise information-driven data sampling and visualization," *Entropy*, vol. 21, no. 7, p. 699, Jul 2019. [Online]. Available: http://dx.doi.org/10.3390/e21070699

[23] T. Rapp, C. Peters, and C. Dachsbacher, "Void-and-cluster sampling of large scattered data and trajectories," *IEEE TVCG*, vol. 26, no. 1, pp. 780–789, 2020.

[24] S. Di, J. Liu *et al.*, "A survey on error-bounded lossy compression for scientific datasets," *arXiv preprint arXiv:2404.02840*, 2024.

[25] B. Harper, "Tropical cyclone parameter estimation in the australian region," *Systems Engineering Australia Pty Ltd for Woodside Energy Ltd, Perth*, vol. 83, 2002.

[26] S. Park, L. Linsen, O. Kreylos, J. Owens, and B. Hamann, "Discrete sibson interpolation," *IEEE TVCG*, vol. 12, no. 2, pp. 243–253, 2006.

[27] L. Bradley, B. Sipőcz, T. Robitaille *et al.*, "astropy/photutils: 1.12.0," Apr. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.10967176

[28] M. D. Buhmann, "Radial basis functions," *Acta Numerica*, vol. 9, p. 1–38, 2000.

[29] S. Sagawa, A. Raghunathan, P. W. Koh, and P. Liang, "An investigation of why overparameterization exacerbates spurious correlations," *arXiv preprint arXiv:2005.04345*, 2020.