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

Category: Research

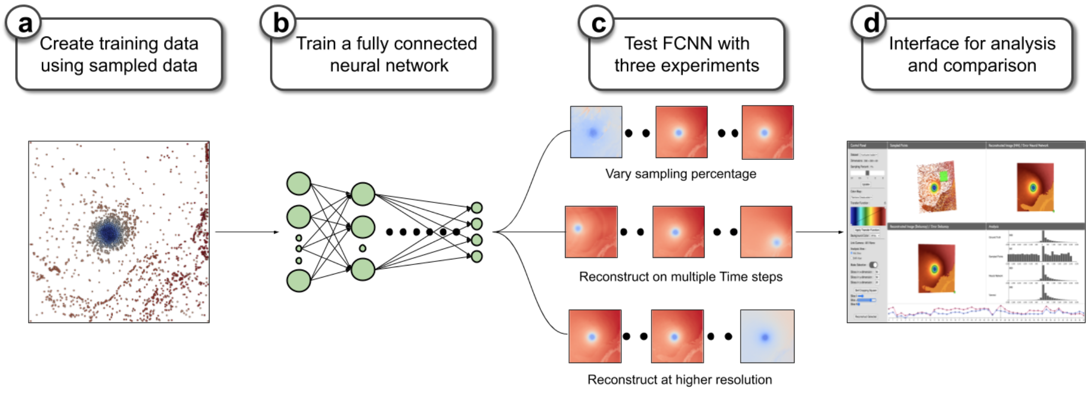

a
Create training data using sampled data
b
Train a fully connected neural network
c
Test FCNN with three experiments
d
Interface for analysis and comparison
Vary sampling percentage
Reconstruct on multiple Time steps
Reconstruct at higher resolution

Figure 1: The high-level workflow of our deep learning-based approach to reconstruct 3D volume datasets from sampled data. (a) After a sampled dataset is generated from a regular grid dataset, (b) a fully connected neural network is trained over the features extracted from the void locations. (c) The network is tested over sampling percentages, an ensemble of such networks is tested over timesteps for a single simulation, and finally the network is tested reconstructing at multiple higher resolutions. (d) We additionally develop an interface to review, analyze, and compare dataset reconstructions between our deep learning approach and Delaunay triangulation.

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 an AI-driven 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, we demonstrate that deep learning-based reconstruction has several advantages, including reconstruction quality and time-to-reconstruct. We propose and evaluate strategies that balance the sampling rates with model training and data reconstruction time to demonstrate how such AI approaches can be tailored for both speed and quality, and develop a visual analytics interface for comparing the reconstruction quality 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

1 Introduction

As high performance computing (HPC) approaches the age of 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 [24], bitmap indexing [21], and adaptive sampling [17]. However, as HPC will soon 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 [6, 7] are addressing this by weighting the importance given to data points when doing sampling. In this way, important features can be preserved by weighting sampling distributions towards 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, Delaunay triangulation [6] is a popular and well-regarded reconstruction algorithm with good signal-to-noise performance. However, a major drawback is its time complexity, which increases exponentially with sampling percent, making it unsuitable for large-scale datasets such as those increasingly found in HPC scientific simulations.

Recently, machine learning (ML) and artificial intelligence (AI) approaches have been proposed for reconstruction [9, 22], employing complex deep learning architectures to predict a high-resolution data from a low resolution data. Thus far, such methods have only been 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, if a dataset is sampled for 1% of its data points, the remaining 99% of data points are discarded. Thus 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 unstructured. Performing convolution or using any machine learning algorithm which takes an image as input is not suitable. The result is that current AI/ML reconstruction approaches are incompatible with the aggressive sampling strategies being developed for exascale computing.

This motivates the work described in this paper: *to develop and test the capabilities of a deep 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 employ fully connected neural networks (FCNNs) for deep learning-based reconstruction of unstructured sampled datasets which goes as an input and a structured reconstructed volume data is produced as an output. Using Delaunay triangulation-based linear interpolation [6] as a baseline for comparison, we test this approach using three well-known scientific simulation benchmark datasets: Hurricane Isabel [1], combustion data [2], and the ExaAM dataset [3] 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 predicting higher resolutions.

Compared to Delaunay triangulation, deep learning has both advantages and disadvantages. The primary drawback is training cost: as the size of a sampled dataset increases the time required to train a model will also increase. However, deep learning holds major advantages. A model once trained can be stored for usage later on (either in subsequent timesteps or simulation runs), meaning the cost of training a neural network is amortized by usage on subsequent timesteps. In contrast, rule-based methods like Delaunay triangulation must reconstruct from scratch at every timestep. Our experimental results show that our FCNNs reconstruct datasets both much faster and at higher quality than Delaunay's triangulation.

Additionally, a key finding in our experiments is that *a neural network trained on a single sampling percentage can effectively reconstruct at different sampling percentages and at different resolutions*. This means that an FCNN can be trained once, and then reconstruct a simulation at different sampling percentages, and can volumetrically upscale the simulation's full resolution. We consistently see across the three studied datasets that neural networks maintain these key features 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 datasets for various timesteps, and resolutions.

More broadly, our experiments indicate that deep learning represents a promising strategy for solving the problem of conversion from unstructured to structured data. To further demonstrate our approach, we develop a novel visual analytics interface for analyzing and comparing dataset reconstructions between FCNNs and Delaunay triangulation. The interface can also help the user identify regions of interest based on high or low quality reconstruction regions to focus their attention on, such as features of interest that are reconstructed poorly. To promote adoption and reuse, this interface codebase is open-sourced. To summarize, the main contribution of our work are threefold:

- We develop a deep learning approach to convert unstructured sampled data to structured volume datasets with higher quality and in faster time compared to the current state-of-the art.
- 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.
- We develop a visualization interface for reconstruction review, comparison, and analysis of reconstructed techniques.

2 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. [4] 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 [9] 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 [22] 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. [11], 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 [5] 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. [13] 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. et al. [15] 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. [10] 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 [25] as TempoGAN which uses a temporal discriminator in addition to a spatial generator which preserves the data's temporal coherence. Weiwel et al. presented [23] 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. [24] proposed a stratified random sampling based algorithm for cosmology simulations to enable interactive post-hoc visualization. Park et al. [18] 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 [16] 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. [8] proposed a point wise mutual information based approach for multivariate sampling to identify regions with high mutual information among the variables. Rapp et al. [19] proposed a method for scattered datasets which extracts a sample of points while preserving its blue noise properties. Biswas et al. [6, 7] 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. [6] technique for all data sampling conducted in this work.

3 Methodology

Figures 1(a–c) 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 deep neural network. We test the capabilities of the neural network by experimenting with (1) different sampling percentages, (2) different timesteps for a single sampling percentage, and (3) different data resolutions. For each experiment, we compare the reconstruction results using deep learning to the reconstruction results using Delaunay triangulation.

In this section, we first introduce the experimental datasets that we use for testing and evaluating deep learning-based reconstruction. 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 6 for more discussion on this).

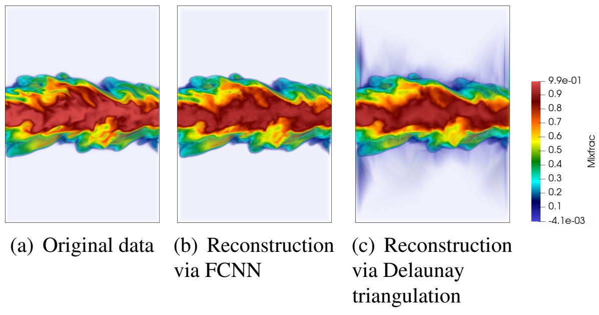

9.9e-01
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
-4.1e-03
Mixfrac
(a) Original data
(b) Reconstruction via FCNN
(c) Reconstruction via Delaunay triangulation

Figure 2: This figure shows reconstruction on the combustion dataset. The original, unsampled data is shown in (a). After sampling 1% of grid points, the dataset is reconstructed using (b) an FCNN and (c) Delaunay triangulation. Reconstruction quality is reported in more detail in Experiment 1.

3.1 Experimental Datasets

We employ three well-known scientific simulation datasets to test deep learning-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 [1] 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 [12]. This dataset has a resolution of $250 \times 250 \times 50$ over 48 timesteps. Figure 1 contains images of the Hurricane Isabel dataset using a blue-white-red diverging color scale. The eye of the hurricane—a very low pressure area—is the blue circle in each sample image.

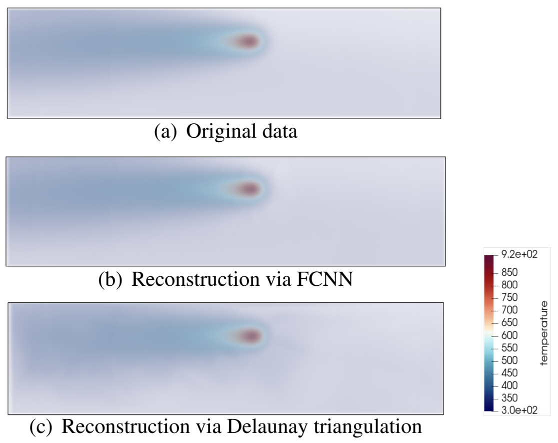


Figure 3: This figure shows reconstruction on the ExaAM dataset. The original, unsampled data is shown in (a). After sampling 1% of grid points, the dataset is reconstructed using (b) an FCNN and (c) Delauany triangulation. Reconstruction quality is reported in more detail in Experiment 1.

Combustion: The combustion dataset [2] 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. This dataset has a resolution of $480 \times 720 \times 120$ over 122 timesteps. Figure 2(a) shows an example of the combustion dataset.

ExaAM: The ExaAM dataset [3] simulates a laser heat source moving through a material—specifically Inconel 625—which first melts and then re-solidifies. It consists of the temperature scalar attribute. We test using the temperature attribute, which becomes very high around the laser's current location. This dataset has a resolution of $20 \times 200 \times 50$ over 108 timesteps. Figure 3(a) shows an example of the ExaAM dataset.

3.2 An Overview of Fully Connected Neural Networks

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}. Edges between layers are associated with weights that determine the amount of importance to be given to a particular neuron.

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. The output layer's values can be considered as the model's end result or prediction. 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*. On the basis of the data, linear or non-linear activation functions can be applied. Forward propagation can be calculated as follows:

$$h^{(l)} = \sigma(\mathbf{W}h^{(l-1)} + b)$$

Here, **W**, h, b, and l represent weights, inputs, bias and the layer number respectively. The σ is the activation applied on each layer. Common activation functions include ReLU, softmax and tanh. In our case, we used ReLU activation, defined as follows:

$$\begin{cases} 0 & h^l < 0 \\ h^l & h^l >= 0 \end{cases}$$

Back Propagation: 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. Common error functions include binary cross-entropy and root mean square error, or RSME. 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 deep learning. Back propagation works computes the error gradients of each neuron, iteratively for each layer from the output layer backwards.

$$\frac{\delta E}{\delta w^{(L)}} = \frac{\delta E}{\delta h^{(L)}} \times \frac{\delta h^{(L)}}{\delta w^{(L)}}$$

Then, an optimization algorithm such as gradient descent updates the weights of the neurons in the model in each layer.

$$w^{(L)} = w^{(L)} - \eta \times \frac{\delta E}{\delta w^{(L)}}$$

where η is the learning rate. Performing this method multiple times results in a decreased loss between the predicted output and the actual output, leading to a trained neural network.

3.3 Training Dataset and FCNN Architecture

During a large-scale scientific simulation, it is reasonable that the full-resolution dataset is only available for the current timestep. Therefore, training data should be confined to a single available timestep. Figure 4 shows the process we use to train an FCNN for scientific data reconstruction.

The dataset's grid points for the timestep are divided into two groups, *void locations* and *sampled points*, 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. The set of void locations for a sampled dataset is the set of points not included in the sampled set. Such void or empty grid locations lack the scalar value which acts like missing data.

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 and (2) the x, y, and z coordinates of the void location itself. The set of vectors for all void locations constitutes the training dataset for the FCNN.

The FCNN's input layer therefore consists of 23 neurons corresponding to the $[1 \times 23]$ vector created for each void point. Five hidden layers are then applied. The output layer predicts a $[1 \times 4]$ vector: the scalar value and the x, y, and z gradients for a grid point.

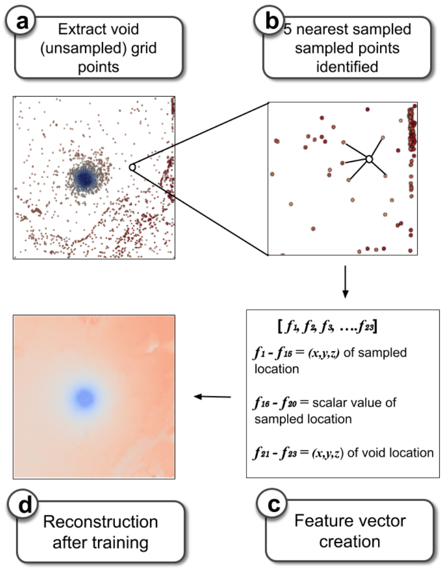

a
Extract void (unsampled) grid points
b
5 nearest sampled sampled points identified
$[f_1, f_2, f_3, \ldots f_{23}]$
$f_1 - f_{15} = (x,y,z)$ of sampled location
$f_{16} - f_{20}$ = scalar value of sampled location
$f_{21} - f_{23} = (x,y,z)$ of void location
c
Feature vector creation
d
Reconstruction after training

Figure 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) (c) The *x,y* and *z* coordinates of each of the sampled point, the coordinates of the corresponding void point and the scalar values associated with each sampled point 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 image as an output.

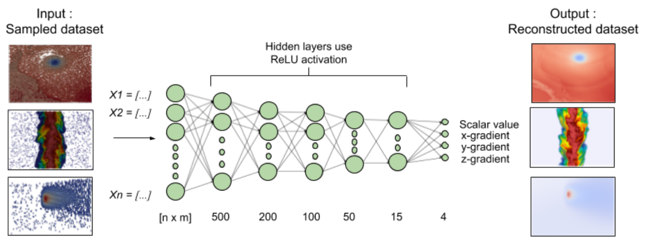


Figure 5: The FCNN architecture that we utilize is composed of five hidden layers of size 500–15, and outputs a scalar value as well as the x-, y-, and z-gradients for a reconstructed point.

3.4 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 disdained 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, 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.

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 [20]

To find the appropriate number of hidden layers, we tested reconstructing the Hurricane Isabel dataset with different numbers of hidden layers (between one and nine hidden layers). Figure 6 shows the results. Reconstruction quality (measured as signal-to-noise ratio, or SNR, see Section 4) is plotted against the number of hidden layers in the FCNN. The SNR for both one hidden layer (20.33) and nine hidden layers (25.71) is lower than for five layers (28.51). Our assumption is that an FCNN with one hidden layer does not learn the features very well because of a high bias (too simple 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.

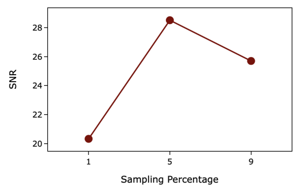


Figure 6: Average SNR values when varying the number of hidden layers on the Hurricane Isabel dataset.

Sampling points. *Sampling percentage* refers to the total percent of data points that are saved in relation to the entire 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 that an FCNN will, when weights are assigned to features, given 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.

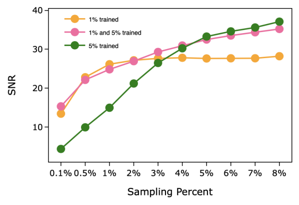


Figure 7: Average SNR values when training on different sampling percentages on the Hurricane Isabel dataset.

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

Gradients in Output Layer: Generating gradient values along with the scalar values in the output layer helps achieve better reconstruction quality as compared to when generating only the scalar values. This is because, by including gradient values the network is enforced to take into consideration the neighbouring locations' values while outputting the scalar value at a given location. For example, two different spatial locations can have the same scalar values, but depending on the shape 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 a study 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 improve the overall reconstruction quality of our network.

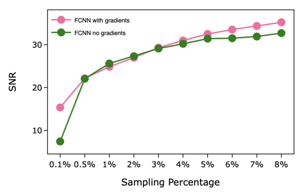


Figure 8: SNR values in gradient vs no gradient in the output layer of the FCNN.

4 Experiments and Results

We test our deep learning-based approach against Delaunay triangulation to reconstruct from sampled data back to a full-resolution dataset. To calculate the reconstruction quality, we use signal-to-noise ratio (SNR), defined as follows:

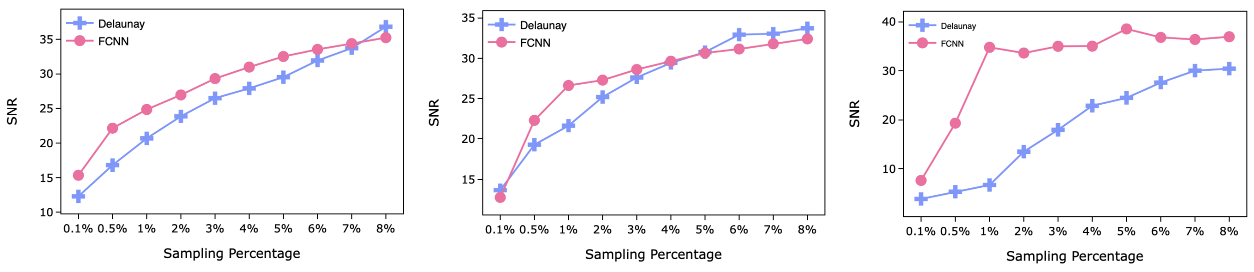
$$SNR = 20 * \log_{10} \frac{\sigma_{raw}}{\sigma_{noise}}$$

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 value, which means the reconstructed image is similar to the original data. Thus if the reconstruction is quite similar to the original data, the variance of the noise present in the reconstructed data decreases, but the variance of the original data will not change. Thus, a well reconstructed image has higher SNR.

In general, for both 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 Delaunay triangulation, but want to see the time required by our deep learning approach to reconstruct the dataset.

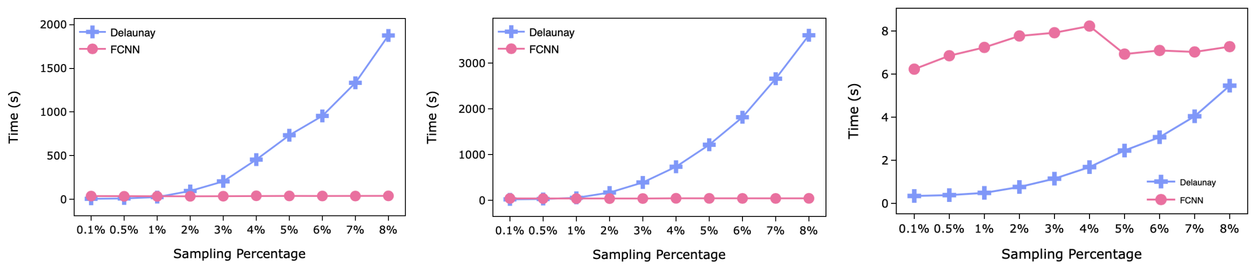
4.1 Testing Setup

We conducted our experiments using the <Anonymous> HPC research cluster at <Anonymous Institution>. This compute cluster contains approximately 500 heterogeneous compute nodes and can compute at over 800 Teraflops. For our experiments, we used a single compute node containing eight CPU cores and one one GeForce GTX 1080 GPU. When conducting experiments, we first use the



(a) Hurricane Isabel (t=25) (b) Combustion ($t = 41$) (c) ExaAM ($t = 78$)

Figure 9: Reconstruction quality (SNR) for FCNN and Delaunay triangulation at different sampling percentages for a single timestep t.



(a) Hurricane Isabel (t=25) (b) Combustion ($t = 41$) (c) ExaAM ($t = 78$)

Figure 10: Reconstruction time in seconds for FCNN and Delaunay trianguation at different sampling percentages for a single timestep t.

sampling technique by Biswas et al. [6] to sample the data for a given sampling percentage (between 0.1% and 8%). This converts the original regular grid dataset, stored as a .vti file (XML Image data), into a point-based dataset stored using the .vtp file format (XML Poly data). Both reconstruction methods are applied on this sampled dataset, and output the reconstructed dataset as .vtp format. To calculate SNR, we compare scalar values in the reconstructred .vtp file to the true scalar values in the original .vti file(s).

4.2 Experiment 1: Varying Sampling Percentages

The first experiment compares reconstruction quality over different sampling percentages at a single timestep. We trained the network as explained in Section 3. We then tested reconstruction on the three datasets with ten sampling percentages from 0.1% to 8%. Figure 9 shows the reconstruction quality and Figure 10 shows the corresponding time to reconstruction.

For both methods, reconstruction quality increases as the sampling percentage increases. At 0.1% sampling percentages, FCNN and Delaunay triangulation have comparable SNR. However, SNR is generally higher for FCNN except for two instance: Hurricane Isabel at 8% and the combustion dataset above 5.0%.

In analyzing reconstruction time, we first note that Figure 10 does not include model training time—this is considered in Experiment 2. For a trained FCNN, reconstruction occurs in constant time (with respect to the full dataset size) and is independent of the sampling percent. In contrast, Delaunay triangulation increases exponentially with regards to the number of data points (i.e., the sampling percentage). This can be seen in all three datasets in Figure 10: the FCNN line is flat while the Delaunay triangulation line arcs upward as sampling percentage increases.

This highlights a key benefit to this experiment: the flexibility of FCNNs. For each dataset, we use a single trained FCNN to reconstruct across all sampling percentages. In contrast, Delaunay triangulation must start from scratch for each reconstruction, and the time increases exponentially based on the number of data points in the sample. For the small ExaAM dataset in Figure 10(c), FCNN's constant time is greater than Delauny triangulation, however, for the two larger datasets reconstruction time via FCNN quickly becomes trivial compared to Delaunay triangulation. As an example, Delaunay triangulation requires about approximately 20 minutes to reconstruct a single timestep for the Hurricane Isabel dataast at 5% sampling percentage. The FCNN reconstructs in 28 seconds. For adaptive techniques, which might vary the number of sampled points at each timestep during a simulation run, deep learning guarantees a fast reconstruction time that Delaunay triangulation cannot.

4.3 Experiment 2: Testing Over Multiple Timesteps

In Experiment 1, FCNNs generally outperform Delaunay triangulation both in terms of quality and reconstruction time. However, Experiment 1 does not consider the time required to train the FCNN models. The reasoning is that, while training time for a neural network is a computationally expensive process even for a straightforward architecture like an FCNN, a key benefit is that a trained neural network can be saved and re-used. In our case, the model can be used over subsequent timesteps or simulation runs. In contrast, Delaunay triangulation reconstructs from scratch at each timestep.

Experiment 2 tests how an FCNN trained on one timestep can be used to reconstruct on other timesteps. We focus on the Hurricane Isabel dataset using a 3% sampling percentage, as this simulation showcases a highly complex weather process with features that change significantly over the course of the run as the hurricane moves across the Gulf of Mexico and makes landfall.

Figures 11 and 12 show reconstruction quality and cumulative reconstruction time over the simulation's 48 timesteps. Each plot contains three lines. The blue line shows reconstruction using Delaunay triangulation. The orange line shows reconstruction using an FCNN trained on timestep $t = 0$ and used to predict on all timesteps. The pink line shows reconstruction using an FCNN that is initially trained on $t = 0$ and then partially retrained every 10th timestep.

The motivation for this partial retraining technique is based the fact that not all data features change in consecutive timesteps. Thus instead of training at each timestep, or fully retraining when reconstruction quality degrades, we instead used an already-trained network to learn the new set of changes in data which occur in the

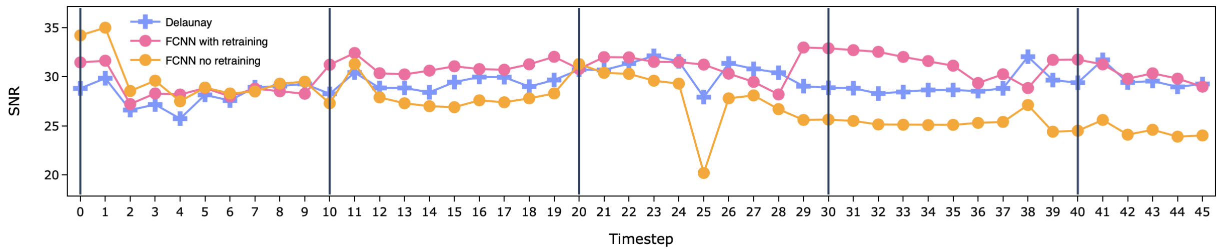


Figure 11: This figure shows reconstruction quality for the Hurricane Isabel dataset over its 48 timesteps using a 3% sampling percentage at each timestep. FCNNs generally provide higher quality than Delaunay triangulation. Vertical lines indicate timesteps where the pink "FCNN with retraining" line was trained ($t = 0$) or retrained (every subsequent 10 timesteps). The orange "FCNN training on $t = 0$" line reconstructs all timesteps using the initial FCNN model trained on the first timestep.

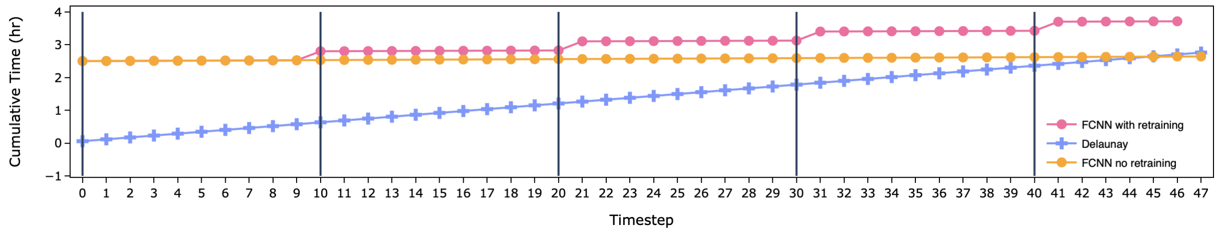


Figure 12: This figure shows the cumulative reconstruction time in hours for the Hurricane Isabel dataset over its 48 timesteps using a 3% sampling percentage at each timestep. At each timestep, reconstruction time using Delaunay triangulation averages 0.05 hours. Vertical lines indicate timesteps where the pink "FCNN with retraining" line was trained ($t = 0$) or retrained (every subsequent 10 timesteps). Reconstruction time for both FCNN methods on timesteps that no retraining happens average 0.16 minutes. When training/re-training time is included, the cumulative times are 3.71 hours for the FCNN with re-training, 2.6 hours for the FCNN trained on the first timestep, and 2.76 hours for Delaunay triangulation.

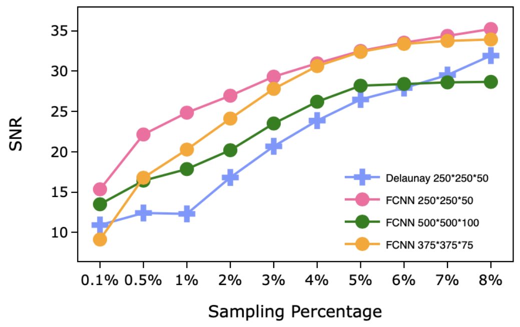
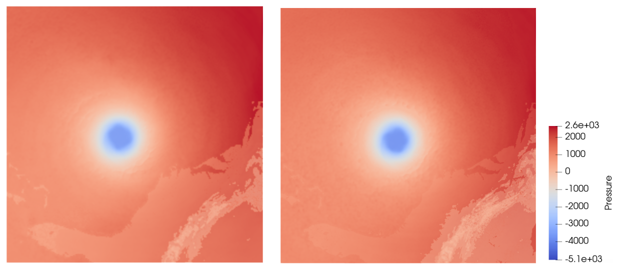


Figure 13: SNR values of Hurricane Isabel dataset at different resolutions at different sampling percentages. Note that Delaunay triangulation did not complete reconstruction (over 12 hours) when run at higher resolutions.



(a) Upscaled reconstruction to $375 \times 375 \times 75$ (b) Upscaled reconstruction to $500 \times 500 \times 100$

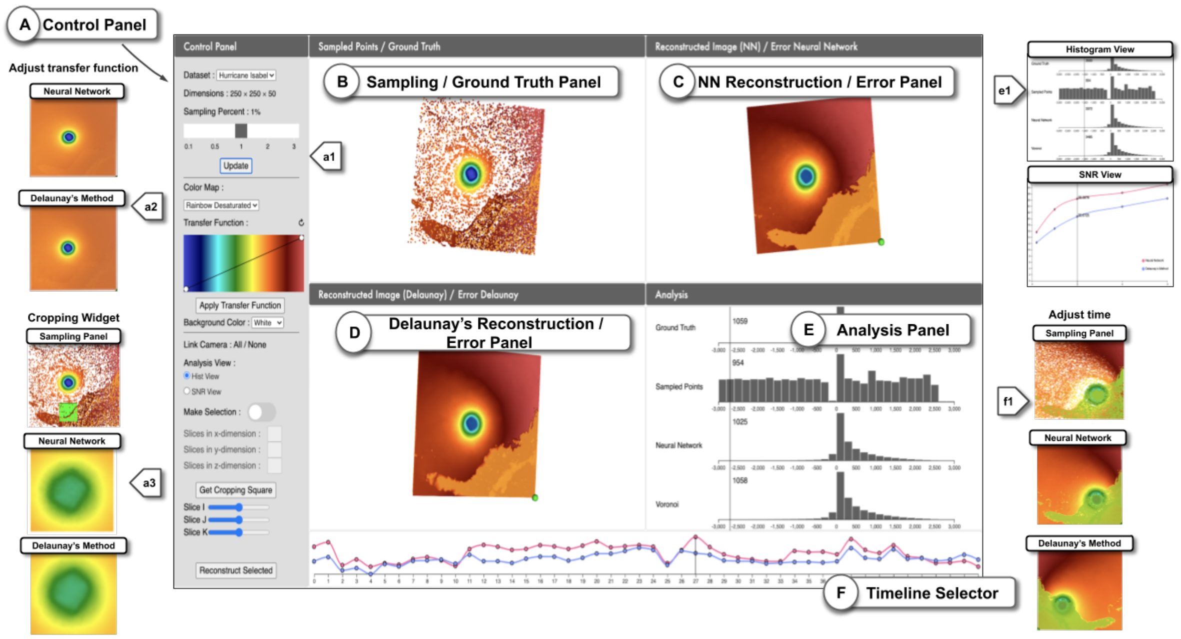
Figure 14: This figure shows volume upscaling while performing reconstruction on the Hurricane Isabel dataset at 4% sampling percentage. The dataset used for sampling and FCNN training is $250 \times 250 \times 50$, but the reconstructed datasets are 1.5× and 2× those dimensions.

next time step by training it for a few epochs till the change in loss starts to become negligible.

In other words, at $t = 0$, an FCNN is fully trained and used to begin reconstructing timesteps.If a certain amount of time has passed (or assuming a more nuanced setup, if the reconstruction quality decreases past a certain threshold), the FCNN can be retrained to a certain epoch, and this updated FCNN is used going forward until retraining is again required.

The advantage to this partial retraining approach is that retraining a network takes much less time than training from scratch, as seen in Figure 12. The training at timestep $t = 0$ required approximately 2.5 hours, but using retrainings cumulatively required 3.71 hours. (Timesteps where retraining occurred are marked by vertical lines). If the FCNN was fully retrained at every timestep, the total time would have equaled approximately 120 hours (2.5 hrs × 48 timesteps). On timesteps that did not require retraining, reconstruction time was negligible for both FCNN methods (averaging 9.6 seconds). In contrast, Delaunay triangulation averaged 3.45 minutes to reconstruct each timestep. Cumulatively, Delaunay triangulation required 2.76 hours to reconstruct all 48 timesteps in the Hurricane Isabel dataset. The FCNN only trained on timestep $t = 0$ took 2.6 hours, and the FCNN that used partial retraining took 3.71 hours.

In terms of reconstruction quality, for a network only trained on the $t = 0$ timestep and tested over other timesteps (with no partial retraining), reconstruction quality decreases as timesteps increase. A possible reason for this is due to a combination of the spatial movement of features of interest (e.g., the hurricane eye) and in physical values throughout the dataset. The FCNN without retraining line in Figure 11 performs worse than Delaunay triangulation for the major-



Dataset : Hurricane Isabel
Dimensions : 250 × 250 × 50
Sampling Percent : 1%
Update
Color Map :
Rainbow Desaturated
Transfer Function :
Apply Transfer Function
Background Color : White
Link Camera : All / None
Analysis View :
● Hist View
○ SNR View
Make Selection :
Slices in x-dimension :
Slices in y-dimension :
Slices in z-dimension :
Get Cropping Square
Slice I
Slice J
Slice K
Reconstruct Selected

Figure 15: The visualization interface consists of six linked panels to support comparing reconstruction quality of FCNN, Delaunay triangulation, and the ground truth data. See Section 5 for details.

ity of timesteps. However, when partial retraining is employed, the FCNN's SNR values are higher compared to Delaunay triangulation for the majority of timesteps. This indicates that partial retraining is a good strategy for maintaining high reconstruction quality.

As a note on this experiment, even though the partial retraining cumulatively took more time than Delaunay triangulation (3.71 vs. 2.76 hours), there are benefits to the deep learning approach. In a simulation with more timesteps, Delaunay triangulation would quickly catch up and overtake the FCNN in cumulative time, even with partial retraining. As training time for a neural network is partially machine dependent, the model (re)training process can be offloaded to optimized compute nodes with higher memory and GPU capabilities, reducing the time required for this step. Additionally, as either the sampling percentage or dataset size increases and Delaunay triangulation's computational time increases exponentially, the trade off between the methods will tilt in favor the deep learning approach (see Experiment 3).

4.4 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 2.

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 3.1, the dataset has a resolution of $250 \times 250 \times 50$. We intended to replicate Experiment 1 with this dataset, but now reconstructing to a double the resolution size: $500 \times 500 \times 100$. This is the resolution of the original Hurricane Isabel dataset, which means we can test reconstruction quality by computing SNR against a resolution that is 2× upscaled on each dimension (and hence a dataset that is 8× larger).

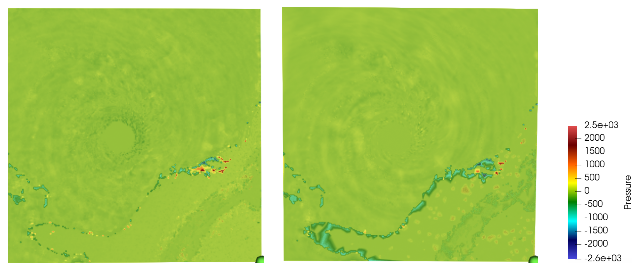
We first tested this 2× upscaled reconstruction using 1% sampling percentage. The Delaunay triangulation process ran for over 12 hours without returning a result. An analysis of Delaunay triangulation's run time indicates why this happened. As Delaunay triangulation is exponential regarding the number of sampled points, doubling the resolution size results in 25M points before sampling. Even at 1% sampling, the method must run on 250k points. We also tested Delaunay triangulation on a 1.5× upscaled resolution ($375 \times 375 \times 75$ points) at 1% sampling percentage, but this calculation again ran for 12 hours without a result. This illustrates a critical limitation for Delaunay triangulation: for large scale simulations, it simply becomes unusable.

We next tested using an FCNN to create an upscaled reconstruction. We use the same FCNN for the Hurricane Isabel dataset from Experiment 1, only now we are reconstructing a dataset double the size that it has been trained on. The 2× upscaling result is shown in Figure 13 (green line) and Figure 13(b). For comparison, Figure 13 also plots the Delaunay triangulation and FCNN results from Experiment 1, to illustrate how upscaling produces a comparable result, as well as a 1.5× upscaled result (again using the same FCNN from Experiment 1). A screenshot of the 1.5× upscaling reconstruction is shown in Figure 14(a). The upscaled reconstruction times averaged 28 seconds for the 2× upscaling and 20 seconds for the 1.5× upscaling, with have SNRs of 30.6 and 26.2, respectively.

A corollary to this experiment is that an FCNN trained on an HPC machine can be downloaded and run on lower-end hardware. For example, we tested reconstruction on a 2018 MacBook Pro running macOS Mojave with 32 GB of memory and an Intel i9 processor. While both FCNN and Delaunay triangulation were able to reconstruct the (non-upscaled) ExaAM dataset at all sampling percentages, as this was the smallest dataset tested, at higher sampling percentages for the Hurricane Isabel and combustion datasets (above 7% and 5%, respectively) the laptop would run out of memory and crash when reconstructing with Delaunay triangulation. In contrast, the FCNN could successfully reconstruct.

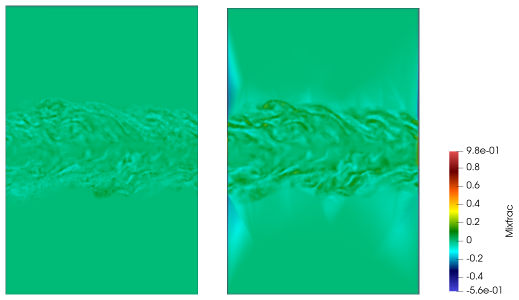
5 Reconstruction Visualization Interface

To further demonstrate and understand the differences between reconstruction via deep learning versus Delaunay triangulation, we developed a visual analytics interface, shown in Figure 15. The intent with this design was to produce a software artifact to help data scientists analyze and understand how the different techniques reconstruct 3D scientific simulation datasets. The interface is primarily



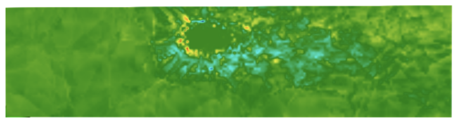
(a) FCNN (b) Delauany triangulation

Figure 16: This figure shows how reconstruction error for the Hurricane Isabel dataset (with 1% sampling percentage) differs between FCNN and Delauany triangulation.

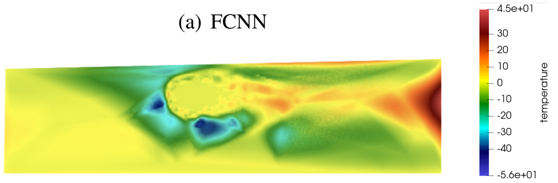


(a) FCNN (b) Delaunay triangulation

Figure 17: This figure shows how reconstruction error for the combustion dataset (with 1% sampling percentage) differs between FCNN and Delaunay triangulation.

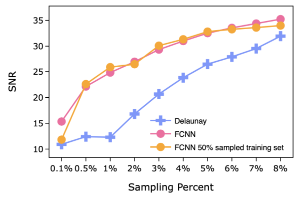


(a) FCNN

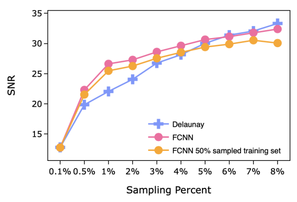


(b) Delaunay triangulation

Figure 18: This figure shows how reconstruction error for the ExaAM dataset (with 1% sampling percentage) differs between FCNN and Delaunay triangulation.



(a) Hurricane Isabel dataset



(b) Combustion

Figure 19: Reconstruction quality comparison between full training set, half training set and Delauany's reconstruction.

written in D3.js and VTK.js, and the codebase is open-sourced at the following URL: Anonymized for submission. (A video demonstrating the interface may be found in supplemental materials.) The interface consists of six linked panels:

The **Control Panel (A)** provides user interactions to load datasets and manipulate settings. Available actions include choosing the dataset (a1), setting the sampling percentage to analyze, setting the colormap, and adjusting the transfer function (a2). For focused analysis and inspection of interesting regions, a cropping functionality is provided, allowing the user to drag and resize a cuboid selector to choose the specific sub-region to reconstruct a3.

The **Sampling / Ground Truth Panel (B)** provides users with a view of the sampled dataset, and on which the cuboid cropping preview is applied (see a2). This panel can be toggled from showing the sampled points to also show the dataset's ground truth—i.e., the original grid point dataset. This panel supports interactive zoom, pan, and rotation.

The **Reconstructed Image Panels (C, D)** show the dataset reconstructed via FCNN and Delaunay triangulation. Like the sampling panel, these panels support interactive zoom, pan, and rotation. Cropping action applied in sampling panel crops the images in these panels (a3). These two panels can be toggled into an error rendering mode, which shows the error between the ground truth and the reconstructed image—Figures 16–18 shows examples for the three datasets. The error rendering can be used to help identify regions with high error compared to regions that are well reconstructed.

The **Analysis Panel (E)** allows the user to toggle two data summarization views. The histogram view (shown in (E)) uses histograms to bin the scalar value distributions in the different dataset representations (original dataset, sampled dataset, reconstruction via FCNN, reconstruction via Delaunay). The user can examine differences histogram bins across datasets to see how the scalar bins differ, particularly to the original dataset (the ground truth). The panel can also toggle to a line chart view (see e1, bottom), which plots SNR over sampling percentages for the current timestep.

Finally, the **Timeline Selector (E)** allows the user to select a timestep to populate panels (B–E). The line chart in this panel shows the SNR values for FCNN and Delaunay triangulation. Clicking on a timestep selects it by moving a vertical bar to that timestep, and updates the corresponding panels (f1),

6 Discussion and Conclusion

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

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. We experimented with reducing this training time further by halving the size of the training set used to build the FCNN. The results, shown in Figure 19, indicate that the decrease in quality (compared to use the full training dataset) was negligible, and even when using 50% of the samples for training, still generally outpaces Delaunay triangulation. As a future work, we plan to formally investigate the idea of intelligent training set creation to support better and faster reconstruction.

FCNN reconstructs better in certain locations compared to Delaunay triangulation. In developing our visual interface to compare reconstruction quality between FCNN and Delaunay triangulation, we noticed that the latter method failed to capture or learn the underlying data dynamics or features of interest. This led us to create the error panel views, which calculates the error at each grid point by taking the difference compared to the ground truth. It is easy to see areas with high error using this view. For example, in Figure 16(b), Delaunay triangulation has trouble reconstructing the coastline around Florida. Similar errors are likewise seen in Figures 17 and 18. In the combustion dataset, Delaunay triangulation has higher error both in the central turbulence regions and around the borders. For the ExaAM dataset, the laser eye is not well reconstructed using Delaunay triangulation. This is one example of how the visual interface helped us qualitatively assess reconstruction quality: even when SNR scores were relatively similar between the two methods, Delaunay triangulation at times had major errors in important features or regions.

Contributions and Limitations. Experiment 3 showed a significant limitation for Delaunay triangulation, as it could not even reconstruct the dataset, while the FCNN could easily do so. The experiments illustrate other key benefits for using deep learning 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 used for reconstruction at different timesteps, sampling percentages, and even dataset resolutions. In addition to these experimental findings, we contribute the process of designing and fine-tuning a well thought-out FCNN architecture (see Section 3) and a visualization interface for comparison and analysis of reconstructed datasets.

Despite these advantages, we also see limitations with the current deep learning approach. The first is training time. Even when partial re-training is used, training time makes up the majority of cumulative reconstruction time. Intelligent set creation provides one strategy to overcome this. A second drawback is dataset specificity. The FCNN is trained on the dataset it reconstructs, while Delaunay triangulation is a general method that can run on any dataset. As a future work, we intend to explore how neural network models can be trained to reconstruct varied simulation datasets. A final potential drawback in our approach (which is also present in Delaunay triangulation) is uncertainty. A neural network that predicts scalar values with a measure of uncertainty can be leveraged to reconstruct the dataset with overall higher SNR values. We additionally plan to explore this as a part of our future work.

7 Conclusion

At current, the use of deep learning for reconstructing point-based scientific datasets is underexplored. To our knowledge, this is the first work to delve into this specific topic. Our results are especially encouraging in two ways: (1) Even using "straightforward" FCNN architectures, we can create an effective neural network architecture that achieves excellent results that are in many ways better than the state-of-the-art reconstruction technique of Delaunay triangulation. (2) The current work establishes a foundation for future opportunities, which will likely greatly improve upon these results by creating much more sophisticated, flexible, and generalizable models.

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 computing simulations. As deep learning increasingly becomes a part of HPC workflows, it is important to keep in mind that explainability and transparency will become key facets to understanding such techniques. Visual interfaces such as the one implemented here showcase a key approach to validate and review such techniques.

References

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

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

[3] https://www.slideshare.net/insideHPC/exa-am-additive-manufacturing-project-for-exascale.

[4]

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

[6] A. Biswas, S. Dutta, E. Lawrence, J. Patchett, J. C. Calhoun, and J. Ahrens. Probabilistic data-driven sampling via multi-criteria importance analysis. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[7] A. Biswas, S. Dutta, J. Pulido, and J. Ahrens. In situ data-driven adaptive sampling for large-scale simulation data summarization. In *Proceedings of the Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*, ISAV '18, page 13–18, New York, NY, USA, 2018. Association for Computing Machinery.

[8] S. Dutta, A. Biswas, and J. Ahrens. Multivariate pointwise information-driven data sampling and visualization. *Entropy*, 21(7):699, Jul 2019.

[9] 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 Pacific Visualization Symposium (PacificVis)*, pages 71–80, 2020.

[10] J. Han, J. Tao, and C. Wang. Flownet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 26(4):1732–1744, 2020.

[11] J. Han and C. Wang. Tsr-tvd: Temporal super-resolution for time-varying data analysis and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):205–215, 2020.

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

[13] 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*, 26(1):23–33, 2020.

[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)*, pages 282–291, 2019.

[15] 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)*, pages 76–85, 2018.

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

[17] B. Nouanesengsy, J. Woodring, J. Patchett, K. Myers, and J. Ahrens. Adr visualization: A generalized framework for ranking large-scale scientific data using analysis-driven refinement. In *2014 ieee 4th symposium on large data analysis and visualization (ldav)*, pages 43–50. IEEE, 2014.

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

[19] T. Rapp, C. Peters, and C. Dachsbacher. Void-and-cluster sampling of large scattered data and trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):780–789, 2020.

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

[21] Y. Su, G. Agrawal, J. Woodring, K. Myers, J. Wendelberger, and J. Ahrens. Taming massive distributed datasets: data sampling using bitmap indices. In *Proceedings of the 22nd international symposium on High-performance parallel and distributed computing*, pages 13–24, 2013.

[22] S. Weiss, M. Chu, N. Thuerey, and R. Westermann. Volumetric isosurface rendering with deep learning-based super-resolution. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2019.

[23] S. Wiewel, M. Becher, and N. Thuerey. Latent space physics: Towards learning the temporal evolution of fluid flow. *Computer Graphics Forum*, 38(2):71–82, 2019.

[24] J. Woodring, J. Ahrens, J. Figg, J. Wendelberger, S. Habib, and K. Heitmann. 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*, pages 1151–1160. Eurographics Association, 2011.

[25] Y. Xie, E. Franz, M. Chu, and N. Thuerey. Tempogan: A temporally coherent, volumetric gan for super-resolution fluid flow. *ACM Trans. Graph.*, 37(4), July 2018.