Accelerated Depth Computation for Surface Boxplots with Deep Learning

Mengjiao Han* SCI Institute
Tushar M. Athawale† Oak Ridge National Laboratory
Jixian Li‡ SCI Institute
Chris R. Johnson§ SCI Institute

ABSTRACT

Functional depth is a well-known technique used to derive descriptive statistics (e.g., median, quartiles, and outliers) for 1D data. Surface boxplots extend this concept to ensembles of images, helping scientists and users identify representative and outlier images. However, the computational time for surface boxplots increases cubically with the number of ensemble members, making it impractical for integration into visualization tools. In this paper, we propose a deep-learning solution for efficient depth prediction and computation of surface boxplots for time-varying ensemble data. Our deep learning framework accurately predicts member depths in a surface boxplot, achieving average speedups of 6X on a CPU and 15X on a GPU for the 2D Red Sea dataset with 50 ensemble members compared to the traditional depth computation algorithm. Our approach achieves at least a 99% level of rank preservation, with order flipping occurring only at pairs with extremely similar depth values that pose no statistical differences. This local flipping does not significantly impact the overall depth order of the ensemble members.

Index Terms: Deep learning, uncertainty visualization, surface boxplot

1 INTRODUCTION

Ensemble simulations, performed in various scientific domains, including hydrodynamics, nuclear science, and climate science, capture the effect of uncertainty in simulation parameters on actual simulations. However, ensemble simulations pose the nontrivial challenge of effectively and efficiently visualizing uncertainty across ensemble members. Over the past few years, various uncertainty visualization techniques (e.g., probabilistic marching cubes [19], contour boxplot [29], surface boxplots [8]) have been developed to summarize and convey important ensemble members to users. A comprehensive overview of ensemble visualization techniques can be found in the survey paper by Wang et al. [27]. In this work, we present an approach using neural networks for computing surface boxplots of time-varying two-dimensional (2D) ensemble data.

The surface boxplots algorithm [8] is an uncertainty visualization method proposed by Genton et al. for generating visual summaries of an ensemble of 2D datasets. The algorithm extends the concept of Tukey's boxplots [25] for 1D data to 2D/3D images, deriving ensemble descriptors (e.g., median member and outlier members) through statistical analysis of the ensemble (Sec. 2.1). In particular, surface boxplots utilize the notion of functional depth [24] to rank ensemble members. Each ensemble member is assigned a rank based on the level of centrality or depth with respect to other ensemble members (Sec. 2.1). However, the cost of computing functional depth increases cubically with the number of ensemble members, making it impractical to implement in visualization tools, especially for time-varying datasets.

In this paper, we present an approach that leverages neural networks to learn the computation of surface boxplots using a subset of time steps from time-varying ensemble data. The trained model is then used to predict the depth order for the remaining time steps. We demonstrate that our method can accurately learn the depth calculation by evaluating the trained model with untrained data from the same simulation models. Our proposed deep learning method provides depth order results that closely match the ground truth and accurately predict median members, which are crucial information from surface boxplots. Additionally, we compare the performance of our method with the traditional surface boxplots computation method to highlight the efficiency and benefits of our approach. Our approach is up to 6 times faster than the original depth computation algorithm with parallel computation using a CPU inference and 15 times faster using a GPU for the larger **Red Sea** dataset [30] with 50 ensemble members. This paper contributes to the field by providing an alternative method for functional depth computation in time-varying datasets, making it feasible for integration into visualization tools.

2 RELATED WORK AND BACKGROUND

2.1 Surface Boxplots for Ensembles

The surface boxplots approach was introduced by Genton et al. [8] for visualization and exploratory analysis of samples of images. This method allows users to detect the most representative sample surface or image and identify potential outlying images, which often contain interesting features not present in most of the images. In their method, they used the notion of volume depth to order the images viewed as surfaces. The member with the highest data depth is considered the most central or median, and other members are ranked outwards from the median based on ordered data depths. We briefly describe the process of computing a surface boxplot for an ensemble dataset, as demonstrated by Genton et al. [8], and introduce the notation used in our paper in supplemental material.

Calculating the modified volume depth (MVD) for an ensemble with n members involves three nested loops: iterating over each member, each grid, and each combination from $C(n-1, j)$ selecting j members from $n-1$. The band depth (BD) or modified band depth (MBD) requires constructing all possible bands, with the computational cost increasing at a rate of $\binom{n}{j}$ for $2 \leq j \leq n$. This process becomes time-consuming as the number of ensemble members and the value of j increase.

2.2 Deep Learning for Visualization

Advances have been made in using machine learning (ML) and deep learning methods for data visualization [5, 10]. Several innovative approaches use deep neural networks for efficient exploration of simulation parameter space via visualization [13, 22], efficient isosurface extraction guided by the uncertainty of implicit neural

*e-mail: mengjiao@sci.utah.edu
†e-mail: athawaletm@ornl.gov
‡jixianli@sci.utah.edu
§crj@sci.utah.edu

representation [16], and uncertainty analysis for super-resolution applications [21]. Han et al. [11] recently proposed the utilization of deep learning models to circumvent costs of expensive Monte Carlo sampling and efficiently predict level-set uncertainty in time-varying ensemble data. Motivated by their approach, we propose the utilization of deep learning models for efficient and reliable data depth prediction for the time-varying ensemble data.

2.3 Uncertainty Visualization

The area of uncertainty visualization has been rapidly evolving over the past couple of decades since data cannot be trusted without understanding of uncertainty. Various survey reports [7, 20, 6, 14] have documented state-of-the-art in uncertainty visualization. Visualizing variations in level-sets [19, 3], direct volume rendering [18, 1], topological features (e.g., critical points [17, 9] and Morse complexes [2]) are a few well-studied methods for analyzing uncertainty across ensemble members. Visualization of uncertainty, however, can lead to cost and memory overhead, which can become a bottleneck in visualization tools and applications. A few recent works proposed novel methods to alleviate cost of uncertainty visualization [11, 28, 4, 15]. In our work, we propose to alleviate cost of computing surface boxplots by using a deep learning approach.

3 Depth Prediction Model for Time-Varying Ensemble Datasets

3.1 Training Data Generation

In our method, for a particular time-varying ensemble datasets, we first normalized the datasets with the range of data values across the entire time-varying ensemble dataset. Then we generated the training data by randomly selecting T time steps from the time-varying ensemble datasets. Let $E = \{S_1, S_2, \ldots, S_n\}$ denote an ensemble of n 2D datasets for each time step. For each member at a given time step, we calculate the depth of data at each grid point using the surface boxplot algorithm described in Section 2.1. Thus, a training sample represents a single grid point of one ensemble member at a given time step, along with the depth for this grid point, resulting in a one-dimensional (1D) vector of size $n+1$.

Figure 1 provides a simple example of our training samples. In this example, there are 3 members, $\{S_1, S_2, S_3\}$, at each time step. Here, $v1_{(0,0)}, v2_{(0,0)}$, and $v3_{(0,0)}$ represent the data values for each member at grid point $(0,0)$. Similarly, $P1_{(0,0)}, P2_{(0,0)}$, and $P3_{(0,0)}$ are the corresponding depth values for each member at grid point $(0,0)$. Each training sample includes the data values followed by the depth value. Notably, the data values for a specific member are moved to the last position before the depth value in each training sample. For example, in S_2 from Figure 1, the data value of $v2_{(0,0)}$ is moved to the last position before the depth value.

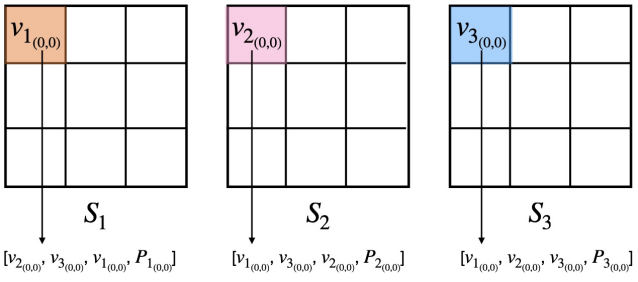

Figure 1: Illustration of training data samples. The figure shows three ensemble members ($\{S_i\}$ where $1 \le i \le 3$) at a single time step. Here, $v_{i_{(x,y)}}$ denotes the data value of ensemble member i at grid (x,y). Each sample includes the data values from the ensemble members along with the depth ($P_{i_{(x,y)}}$), which is used for loss computation during the training process.

3.2 Neural Network

We adapt the network architecture proposed by Han et al. [11, 12], which consists of a latent encoder E and a latent decoder D, both constructed with multilayer perceptrons (MLP), specifically a series of fully connected (FC) layers (Figure 2).

The data values from ensembles, referred to as *ensemble_data*, serve as the input for the encoder, which comprises a series of FC layers. The dimension of the input corresponds to the number of ensembles n. The encoded latent vectors are subsequently fed into another series of FC layers for decoding to the depth. The sinusoidal activation function, as demonstrated by Sitzmann et al. [23], has proven to be more accurate and faster. Thus, we adopted the sinusoidal activation function after each FC layer, except for the last layer of the latent decoder D. To ensure the output values remain within the [0, 1] range, we applied the Sigmoid activation function before the output layer.

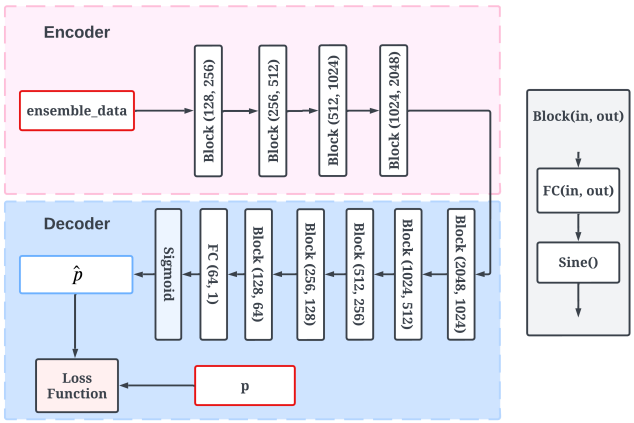

Figure 2: Illustration of network architecture. The network in our method is constructed using multilayer perceptrons (MLP) with a sinusoidal activation function. A sigmoid activation function is applied in the final layer to ensure the output is between $[0,1]$. It takes the data values from the ensemble members and the target depth p as inputs, and outputs the predicted depth ($\hat{p}$). The data values are used to predict the depth $\hat{p}$, while the target depth p is used to compute the loss.

4 Experiments and Results

In this section, we evaluated the network performance (Sec. 4.2) and compared our proposed deep learning-based approach with the original surface boxplots computation algorithm [8] (Sec. 4.3). We demonstrated that our proposed method is both accurate and efficient by analyzing time-varying ensemble datasets from the IRI/LDEO Climate Data Library[1] and the Red Sea simulations performed at the KAUST Supercomputing Lab[2] (Sec. 4.1). We used $j = 2$ for computing depth in generating the training data. The models were trained on dual RTX 3090 GPUs and evaluated on a desktop equipped with an Intel(R) Xeon(R) E5-2640 CPU (40 cores, 128GB memory) and one NVIDIA Titan RTX GPU.

4.1 Data Sets

The **Wind** dataset is from the ECMWF Sub-seasonal to Seasonal (S2S) Prediction Project [26]. The *pressure_level_wind* dataset, obtained using the NECP ensemble forecast system, forms an ensemble with 15 members. We used the *U_Component_Wind* ensemble at a pressure level of 200 HPA, with data from January 1, 2015, and a forecast period of 45 days. The spatial domain covers from 0°E to 1.5°W in longitude and from 90°N to 90°S in latitude, with a grid size of $[240 \times 121]$. The **Red Sea** dataset [30] comprises ensemble

[1] http://iridl.ldeo.columbia.edu/
[2] https://kaust-vislab.github.io/SciVis2020/

simulations of variables relevant to oceanography, such as velocity and temperature. These simulations were performed over a domain with a spatial resolution of $500 \times 500 \times 50$ for 60 time steps. For our experiment, we analyzed the depth order of temperature across 50 ensemble members corresponding to the ocean surface (the top 2D data slice).

Datasets	#Time Steps	#Training Samples	Training (hr)	Model Size (MB)
Wind	10	4.36M	1.28	55.25
Red Sea	30	54M	11.20	55.25

Table 1: Training time and model size for the **Wind** and **Red Sea** datasets. *#TimeSteps* indicates the number of time steps used for training. *#TrainingSamples* represents the number of training samples. We observed that the training time is linearly proportional to the number of training samples. Storing a model configured with 4 encoder layers, 6 decoder layers, and a latent vector dimension of 2048 requires 55.25 MB.

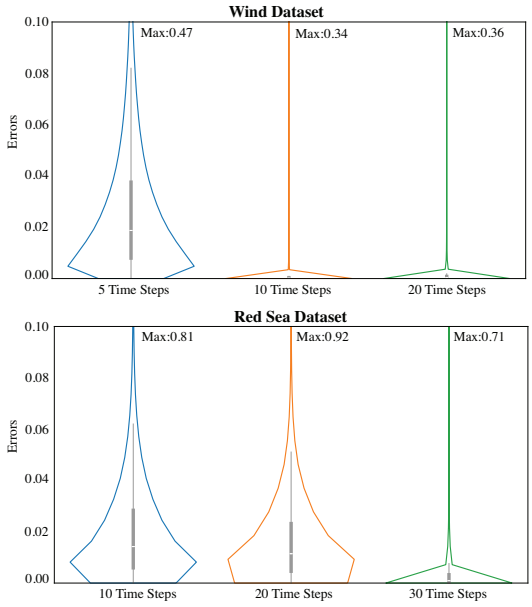

Figure 3: Violin plots of absolute errors between the depth predicted by the neural network and the ground truth. The errors are computed per grid. Increasing the number of time steps used for training reduces these errors. For the **Wind** dataset, using 10 time steps significantly improves the network's performance. For the **Red Sea** dataset, 30 time steps yield the best accuracy. More complex datasets with more ensemble members, such as the **Red Sea**, require more extensive training to achieve optimal performance.

4.2 Model Performance

4.2.1 Training Performance and Model Size

To benchmark the training performance, we randomly selected 10 time steps to generate the training datasets for the **Wind** datasets and 30 time steps for the **Red Sea** datasets, as it has three times more ensemble members. We evaluated our study using the remaining data. We trained the model for 100 epochs. The number of training samples, training time, and the size of the model for each dataset are shown in Table 1. From the results, we observe that the training time is linear to the number of training samples. The trained model needs 55.25 MB of storage with 4 encoder layers, 6 decoder layers, and a 2048-dimensional latent vector.

4.2.2 Prediction Accuracy

To evaluate the impact of the number of training samples on the performance of our model quantitatively, we increased the number of time steps used in the training process and recorded the absolute errors compared to the ground truth depth per grid point for each ensemble member (Figure 3). We randomly selected 5, 10, and 15 time steps for the **Wind** dataset and 10, 20, and 30 time steps for the **Red Sea** dataset.

#Time Step		Depth Order	Level of Rank Preservation
#8	GT	[13, 14, 2, 1, 12, 5, 6, 9, 11, 8, 3, 7, 10, 4, 0]	100%
	Pred	[13, 14, 2, 1, 12, 5, 6, 9, 11, 8, 3, 7, 10, 4, 0]	
#9	GT	[14, 13, 2, 6, 8, 12, 5, 1, 9, 7, 11, 3, 10, 0, 4]	99.04%
	Pred	[14, 13, 2, 6, 8, 12, 5, 1, 7, 9, 11, 3, 10, 0, 4]	
#11	GT	[14, 2, 6, 1, 8, 9, 13, 3, 5, 11, 10, 12, 4, 7, 0]	100%
	Pred	[14, 2, 6, 1, 8, 9, 13, 3, 5, 11, 10, 12, 4, 7, 0]	
#19	GT	[1, 12, 9, 3, 11, 8, 2, 7, 4, 10, 5, 13, 6, 14, 0]	100%
	Pred	[1, 12, 9, 3, 11, 8, 2, 7, 4, 10, 5, 13, 6, 14, 0]	
#21	GT	[2, 3, 12, 9, 7, 1, 0, 13, 14, 10, 8, 11, 5, 4, 6]	100%
	Pred	[2, 3, 12, 9, 7, 1, 0, 13, 14, 10, 8, 11, 5, 4, 6]	
#26	GT	[2, 12, 9, 1, 3, 8, 14, 0, 6, 7, 10, 11, 13, 4, 5]	100%
	Pred	[2, 12, 9, 1, 3, 8, 14, 0, 6, 7, 10, 11, 13, 4, 5]	
#30	GT	[2, 12, 1, 14, 6, 8, 3, 10, 7, 9, 5, 0, 11, 13, 4]	100%
	Pred	[2, 12, 1, 14, 6, 8, 3, 10, 7, 9, 5, 0, 11, 13, 4]	
#31	GT	[12, 1, 11, 2, 6, 0, 14, 7, 5, 9, 10, 3, 8, 13, 4]	100%
	Pred	[12, 1, 11, 2, 6, 0, 14, 7, 5, 9, 10, 3, 8, 13, 4]	
#40	GT	[7, 11, 6, 2, 4, 10, 3, 12, 9, 14, 5, 1, 0, 8, 13]	100%
	Pred	[7, 11, 6, 2, 4, 10, 3, 12, 9, 14, 5, 1, 0, 8, 13]	
#41	GT	[7, 6, 10, 4, 11, 9, 2, 12, 14, 3, 8, 1, 5, 13, 0]	99.04%
	Pred	[7, 6, 10, 11, 4, 9, 2, 12, 14, 3, 8, 1, 5, 13, 0]	

Table 2: Comparison of neural network predicted (Pred) depth order to the ground truth (GT) shows the depth order increasing from left to right. The rank preservation level indicates the inversion ratio of Pred to GT, with incorrect predictions highlighted in orange. The neural network, trained on 10 time steps, accurately predicts the depth order for most time steps in the **Wind** dataset and correctly predicts the median ensemble member. Occasional order flips occur when depths are very similar. For example, in time step #9, the depths of ensemble members #9 and #7 are 0.3389126 and 0.33892963, differing by about 0.000017. Local shuffles from prediction inaccuracies do not significantly affect the overall ranking, as shown by the rank preservation level in the rightmost column.

As observed from Figure 3, the results depicted in the violin plots indicate that training with 10 time steps significantly reduces errors compared to training with 5 time steps for the **Wind** dataset. Additionally, using more than 10 time steps does not provide substantial improvements, suggesting that 10 time steps may be sufficient for this dataset. For the **Red Sea** dataset, the errors decrease significantly when increasing the number of time steps from 20 to 30, with 30 time steps providing the best results. This indicates a stronger dependency on a larger number of time steps for accurate modeling in the **Red Sea** dataset compared to the **Wind** dataset, likely due to the **Red Sea** dataset having more than three times the number of ensemble members and about eight times larger in resolution.

These findings from Figure 3 underscore the importance of selecting an adequate number of training samples for different datasets. While the **Wind** dataset achieves satisfactory performance with a relatively smaller number of time steps, the **Red Sea** dataset benefits significantly from a larger number of time steps. The results suggest that datasets with greater complexity and more ensemble members, like the **Red Sea**, require more extensive training to achieve optimal performance. Conversely, simpler datasets with fewer ensemble members, such as the **Wind** dataset, can reach satisfactory performance with fewer time steps.

Tables 2 and 3 show the computed depth order using the trained neural network compared to the ground truth. In our experiments, we denote the ground truth depth order as the natural order and count the inversions[3] in the predicted depth order. We then use the number of inversions divided by the total number of inversions of the depth order to indicate the level of rank preservation, where a ratio of 100% indicates an exact accurate prediction. Based on our findings in Figure 3, we used 10 time steps for training the network for the **Wind** dataset and 30 time steps for the **Red Sea** dataset.

For the **Wind** dataset, the trained neural network can compute the depth order exactly accurately to the ground truth for most time steps, achieving at least a 0.9904 level of rank preservation with

[3] https://en.wikipedia.org/wiki/Inversion_(discrete_mathematics)

#Time Step		Depth Order	Level of Rank Preservation
#3	GT	[35, 9, 5, 39, 24, 29, 20, 34, 44, 22, 14, 4, 31, 42, 18, 49, 1, 33, 7, 45, 46, 16, 28, 19, 6, 48, 13, 43, 21, 37, 2, 47, 25, 0, 8, 40, 15, 30, 10, 11, 36, 32, 26, 27, 17, 12, 3, 38, 41, 23]	99.84%
	Pred	[35, 9, 5, 39, 24, 29, 20, 34, 44, 22, 14, 4, 31, 42, 18, 49, 1, 33, 7, 45, 46, 16, 28, 19, 48, 6, 13, 43, 21, 37, 2, 47, 25, 0, 8, 40, 15, 30, 10, 11, 36, 32, 26, 17, 27, 12, 3, 38, 41, 23]	
#9	GT	[9, 5, 39, 29, 35, 20, 24, 44, 42, 4, 14, 49, 34, 18, 31, 22, 33, 16, 28, 46, 45, 48, 21, 43, 19, 47, 13, 1, 7, 6, 30, 0, 8, 37, 25, 11, 40, 27, 3, 15, 26, 12, 10, 2, 38, 41, 36, 32, 17, 23]	99.92%
	Pred	[9, 5, 39, 29, 35, 20, 24, 44, 42, 4, 14, 49, 34, 18, 31, 22, 33, 16, 28, 46, 45, 48, 21, 43, 47, 19, 13, 1, 7, 6, 30, 0, 8, 37, 25, 11, 40, 27, 3, 15, 26, 12, 10, 2, 38, 41, 36, 32, 17, 23]	
#17	GT	[35, 29, 5, 9, 39, 20, 14, 4, 44, 42, 24, 49, 22, 18, 47, 28, 43, 16, 13, 48, 46, 21, 45, 33, 31, 34, 8, 30, 7, 11, 25, 27, 19, 12, 1, 6, 41, 26, 38, 0, 37, 40, 3, 10, 15, 2, 32, 23, 36, 17]	99.92%
	Pred	[35, 29, 5, 9, 39, 20, 14, 4, 44, 42, 24, 49, 22, 18, 47, 28, 43, 16, 13, 48, 46, 21, 45, 33, 31, 34, 8, 30, 7, 11, 25, 27, 19, 12, 6, 1, 41, 26, 38, 0, 37, 40, 3, 10, 15, 2, 32, 23, 36, 17]	
#19	GT	[35, 29, 5, 9, 20, 39, 4, 42, 14, 44, 24, 49, 22, 18, 28, 47, 43, 48, 13, 16, 21, 45, 31, 46, 34, 8, 30, 11, 33, 7, 27, 12, 25, 1, 10, 6, 37, 38, 0, 41, 19, 26, 40, 3, 15, 2, 32, 23, 36, 17]	99.76%
	Pred	[35, 29, 5, 9, 20, 39, 4, 42, 44, 14, 24, 49, 22, 18, 47, 28, 43, 48, 13, 16, 21, 45, 31, 46, 34, 8, 30, 11, 33, 7, 27, 12, 25, 10, 1, 6, 37, 38, 0, 41, 19, 26, 40, 3, 15, 2, 32, 23, 36, 17]	
#22	GT	[35, 29, 9, 20, 39, 5, 4, 42, 44, 14, 24, 18, 49, 47, 28, 43, 22, 16, 21, 48, 13, 46, 45, 34, 8, 7, 30, 11, 31, 33, 27, 12, 25, 37, 38, 1, 41, 0, 10, 3, 19, 40, 6, 26, 15, 2, 32, 23, 36, 17]	99.76%
	Pred	[35, 9, 29, 20, 39, 5, 4, 42, 44, 14, 24, 18, 49, 47, 28, 43, 22, 16, 21, 48, 13, 46, 45, 34, 8, 7, 30, 11, 31, 33, 27, 12, 25, 37, 38, 1, 41, 10, 0, 3, 40, 19, 6, 26, 15, 2, 32, 23, 36, 17]	
#30	GT	[20, 35, 29, 9, 5, 4, 39, 42, 44, 14, 49, 47, 18, 28, 24, 43, 21, 13, 22, 16, 48, 46, 8, 45, 11, 34, 27, 31, 30, 12, 33, 7, 38, 25, 41, 37, 1, 0, 3, 10, 6, 26, 40, 19, 15, 23, 32, 36, 2, 17]	99.67%
	Pred	[20, 35, 29, 9, 5, 4, 39, 42, 44, 14, 49, 18, 47, 28, 24, 43, 21, 13, 22, 16, 48, 46, 8, 45, 11, 34, 27, 31, 30, 12, 33, 7, 38, 25, 37, 41, 0, 1, 3, 10, 6, 40, 26, 19, 15, 23, 32, 36, 2, 17]	
#36	GT	[35, 29, 20, 5, 4, 9, 39, 42, 44, 47, 14, 21, 49, 43, 24, 28, 18, 13, 22, 16, 48, 8, 11, 46, 27, 45, 30, 34, 12, 33, 7, 31, 41, 38, 37, 0, 25, 1, 10, 26, 3, 6, 19, 40, 15, 23, 36, 32, 2, 17]	99.92%
	Pred	[35, 29, 20, 5, 4, 9, 39, 42, 44, 47, 14, 21, 49, 43, 24, 28, 18, 22, 13, 16, 48, 8, 11, 46, 27, 45, 30, 34, 12, 33, 7, 31, 41, 38, 37, 0, 25, 1, 10, 26, 3, 6, 19, 40, 15, 23, 36, 32, 2, 17]	
#42	GT	[35, 4, 29, 5, 20, 42, 9, 39, 44, 47, 21, 49, 14, 28, 43, 24, 13, 22, 18, 16, 8, 48, 11, 46, 30, 27, 45, 12, 38, 37, 7, 33, 41, 34, 31, 1, 25, 40, 26, 10, 0, 6, 3, 23, 15, 19, 36, 32, 2, 17]	99.84%
	Pred	[35, 4, 29, 5, 20, 42, 9, 39, 44, 47, 21, 49, 14, 28, 43, 24, 13, 22, 18, 16, 48, 8, 11, 46, 30, 27, 45, 12, 38, 37, 7, 33, 34, 41, 31, 1, 25, 40, 26, 10, 0, 6, 3, 23, 15, 19, 36, 32, 2, 17]	
#47	GT	[29, 4, 35, 5, 20, 42, 9, 44, 39, 21, 47, 14, 49, 43, 28, 13, 22, 24, 18, 16, 48, 8, 11, 46, 30, 12, 27, 45, 38, 7, 31, 37, 41, 33, 25, 34, 0, 40, 26, 1, 3, 10, 6, 23, 15, 19, 32, 36, 2, 17]	99.76%
	Pred	[29, 4, 35, 20, 5, 42, 9, 44, 39, 21, 47, 14, 49, 43, 28, 13, 22, 24, 18, 16, 48, 8, 11, 46, 30, 12, 27, 45, 38, 7, 37, 31, 41, 33, 25, 34, 40, 0, 26, 1, 3, 10, 6, 23, 15, 19, 32, 36, 2, 17]	
#56	GT	[29, 4, 42, 20, 5, 44, 35, 21, 9, 47, 39, 14, 49, 28, 43, 22, 18, 16, 13, 24, 8, 11, 46, 48, 30, 27, 45, 7, 41, 38, 37, 12, 33, 0, 31, 25, 40, 3, 26, 1, 23, 34, 10, 15, 6, 19, 2, 36, 32, 17]	99.67%
	Pred	[29, 4, 42, 20, 5, 44, 35, 21, 9, 47, 39, 14, 49, 28, 43, 22, 18, 16, 13, 8, 24, 11, 46, 48, 30, 27, 45, 7, 41, 38, 37, 12, 33, 0, 31, 25, 40, 3, 23, 26, 1, 34, 10, 15, 6, 19, 2, 36, 32, 17]	

Table 3: Comparison of neural network predicted (Pred) depth order to the ground truth (GT) shows the depth order increasing from left to right. The rank preservation level indicates the inversion ratio of Pred to GT, with incorrect predictions highlighted in orange. The neural network, trained on 30 time steps, predicts the depth order with at least 84% accuracy (42 out of 50) and correctly identifies the median ensemble member. Occasional order flips occur when depths are very similar. For example, in time step #3, the depths of ensemble members #6 and #48 are 0.34113405 and 0.34136669, differing by about 0.00023. Local shuffles from prediction inaccuracies do not significantly affect the overall ranking, as shown by the rank preservation level in the rightmost column.

one flip in the order. The predicted results for the **Red Sea** dataset achieved a rank preservation level ranging from 0.9967 to 0.9992. An important aspect of depth order computation is the identification of the median ensemble member. As shown in Tables 2 and 3, the predicted median ensemble members are all correct.

Moreover, although there are occasional order flips in our predictions, these occur when the depths are very similar. For instance, in time step #3 of the **Red Sea** dataset (Table 3), the depth of ensemble member #6 is 0.34113405, and the depth of ensemble member #48 is 0.34136669. The difference between these two depths is approximately 0.00002. Similarly, in time step #9 of the **Wind** dataset, the depth of ensemble member #9 is 0.3389126, and the depth of ensemble member #7 is 0.33892963. The difference between these two depths is approximately 0.00001. These minimal differences suggest that the neural network's occasional order flips are not significant and are within an acceptable error margin, especially considering the precision required for such fine distinctions. Further, minor local shuffles arising from prediction inaccuracies do not significantly affect the representative nature of ensemble members (e.g., median, outliers) with respect to the entire ensemble.

4.3 Computational Performance and Comparison

In Table 4, we compare the computation time of the traditional depth computation algorithm with parallel computation to the performance of our deep learning method for depth prediction. We implemented the traditional depth computation algorithm using C++ and parallelized it with the TBB[4] library. To compare our neural network's performance, we evaluated it on GPU and CPU using the same testing datasets and set the batch size to 3000.

Datasets	#Ensembles	Traditional Approach (CPU)	DL (CPU)	DL (GPU)
Wind	15	0.336 s	11.50 s	5.85 s
Red Sea	50	245.81 s	40.66 s	16.61 s

Table 4: Comparison of computational time between the traditional depth computation (Traditional Approach) and our deep-learning-based approach (DL). The traditional approach uses C++ with the TBB library, while the DL approach uses PyTorch with a batch size of 3000. Our method achieves up to 15x speed-up with GPU inference and 6x with CPU inference for the **Red Sea** dataset. However, for the smaller **Wind** dataset, our neural network is slower than the traditional algorithm.

By comparing the computational time to the parallel version of the depth computation, as shown in Table 4, our deep learning method provides a speed-up of up to 15X faster using GPU inference and 6X faster using CPU inference for the **Red Sea** dataset. However, for the smaller **Wind** dataset, our neural network is slower than the traditional depth computation algorithm.

These results highlight the efficiency of our deep learning approach for large and complex datasets like the **Red Sea**, where the high number of ensemble members and larger spatial domain benefit significantly from GPU acceleration. The neural network's ability to parallelize operations and leverage GPU capabilities allows for substantial time savings in such scenarios. However, when taking the training time into consideration, the deep learning-based approach is not as efficient as the traditional computation. The initial time investment required for training the neural network can offset the gains achieved during inference, especially for smaller datasets or less complex tasks.

5 Conclusion

We propose a deep neural network to predict the depth order in computing surface boxplots for time-varying scalar ensemble data. This study is the first to apply deep learning to functional depth computation. Our results show that the model's depth order predictions closely match the ground truth and accurately identify median members, which are essential for surface boxplots. Our method is up to 6 times faster than the original depth computation algorithm when using a CPU for inference and 15 times faster when using a GPU for the **Red Sea** dataset with 50 ensemble members. Despite the long training times typical of deep learning projects, the fast inference speed for larger and more complex datasets makes our neural network approach well-suited for integration into visualization tools to quickly compute surface boxplots.

In the future, we plan to explore the model's performance on other datasets with varying complexity and ensemble sizes to gain deeper insights into its generalizability and potential areas for enhancement. We also aim to optimize the training process to reduce time and resource requirements, making the approach more practical. Moreover, future work could focus on improving the precision of the model to further reduce minor discrepancies. Additionally, integrating the neural network into surface boxplot visualization tools would be another valuable application. Finally, we will investigate the applicability of transfer learning [31] using our method on other ensemble data from the same simulation, thereby broadening the scope of its utility.

[4] https://www.intel.com/content/www/us/en/developer/tools/oneapi/onetbb.html

ACKNOWLEDGMENTS

This work was partially supported by the Intel OneAPI CoE, the Intel Graphics and Visualization Institutes of XeLLENCE, and the DOE Ab-initio Visualization for Innovative Science (AIVIS) grant 2428225. Additionally, this work was partially supported in part by the U.S. Department of Energy (DOE) RAPIDS-2 SciDAC project under contract number DE-AC0500OR22725.

REFERENCES

[1] T. M. Athawale, B. Ma, E. Sakhaee, C. R. Johnson, and A. Entezari. Direct volume rendering with nonparametric models of uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1797–1807, Feb. 2021. doi: 10.1109/TVCG.2020.3030394 2

[2] T. M. Athawale, D. Maljovec, L. Yan, C. R. Johnson, V. Pascucci, and B. Wang. Uncertainty visualization of 2D Morse complex ensembles using statistical summary maps. *IEEE Transactions on Visualization and Computer Graphics*, 28(4):1955–1966, Apr. 2022. doi: 10.1109/TVCG.2020.3022359 2

[3] T. M. Athawale, S. Sane, and C. R. Johnson. Uncertainty visualization of the marching squares and marching cubes topology cases. In *2021 IEEE Visualization Conference (VIS)*, pp. 106–110, 2021. doi: 10.1109/VIS49827.2021.9623267 2

[4] T. M. Athawale, Z. Wang, C. R. Johnson, and D. Pugmire. Data-Driven Computation of Probabilistic Marching Cubes for Efficient Visualization of Level-Set Uncertainty. In C. Tominski, M. Waldner, and B. Wang, eds., *EuroVis 2024 - Short Papers*. The Eurographics Association, 2024. doi: 10.2312/evs.20241071 2

[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, 2018. 1

[6] G. Bonneau, H. Hege, C. R. Johnson, M. Oliveira, K. Potter, P. Rheingans, and T. Schultz. Overview and state-of-the-art of uncertainty visualization. In M. Chen, H. Hagen, C. Hansen, C. R. Johnson, and A. Kauffman, eds., *Scientific Visualization: Uncertainty, Multifield, Biomedical, and Scalable Visualization*, pp. 3–27. Springer London, 2014. doi: 10.1007/978-1-4471-6497-5_1 2

[7] K. Brodlie, R. A. Osorio, and A. Lopes. A review of uncertainty in data visualization. In J. Dill, R. Earnshaw, D. Kasik, J. Vince, and P. C. Wong, eds., *Expanding the Frontiers of Visual Analytics and Visualization*, pp. 81–109. Springer Verlag London, 2012. 2

[8] M. G. Genton, C. Johnson, K. Potter, G. Stenchikov, and Y. Sun. Surface boxplots. *Stat*, 3(1):1–11, 2014. doi: 10.1002/sta4.39 1, 2

[9] D. Günther, J. Salmon, and J. Tierny. Mandatory critical points of 2D uncertain scalar fields. *Computer Graphics Forum*, 33(3):31–40, July 2014. doi: 10.1111/cgf.12359 2

[10] J. Han and C. Wang. Coordnet: Data generation and visualization generation for time-varying volumes via a coordinate-based neural network. *IEEE Transactions on Visualization and Computer Graphics*, 29(12):4951–4963, 2022. 1

[11] M. Han, T. M. Athawale, D. Pugmire, and C. R. Johnson. Accelerated probabilistic marching cubes by deep learning for time-varying scalar ensembles. In *2022 IEEE Visualization and Visual Analytics (VIS)*, pp. 155–159. IEEE, 2022. 2

[12] M. Han, J. Li, S. Sane, S. Gupta, B. Wang, S. Petruzza, and C. R. Johnson. Interactive visualization of time-varying flow fields using particle tracing neural networks. In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pp. 52–61. IEEE, 2024. 2

[13] W. He, J. Wang, H. Guo, K. Wang, H. Shen, M. Raj, Y. G. Nashed, and T. Peterka. InSituNet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 26(01):23–33, jan 2020. doi: 10.1109/TVCG.2019.2934312 1

[14] A. Kamal, P. Dhakal, A. Y. Javaid, V. K. Devabhaktuni, D. Kaur, J. Zaientz, and R. Marinier. Recent advances and challenges in uncertainty visualization: a survey. *Journal of Visualization*, 24(5):861–890, May 2021. doi: 10.1007/s12650-021-00755-1 2

[15] H. Li, I. J. Michaud, A. Biswas, and H. Shen. Efficient level-crossing probability calculation for gaussian process modeled data. In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pp. 252–261. IEEE Computer Society, Los Alamitos, CA, USA, apr 2024. doi: 10.1109/PacificVis60374.2024.00035 2

[16] H. Li and H.-W. Shen. Improving efficiency of iso-surface extraction on implicit neural representations using uncertainty propagation. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–13, 2024. doi: 10.1109/TVCG.2024.3365089 2

[17] T. Liebmann and G. Scheuermann. Critical points of Gaussian-distributed scalar fields on simplicial grids. *Computer Graphics Forum*, 35(3):361–370, 2016. doi: 10.1111/cgf.12912 2

[18] S. Liu, J. A. Levine, P.-T. Bremer, and V. Pascucci. Gaussian mixture model based volume visualization. In *IEEE Symposium on Large Data Analysis and Visualization (LDAV)*, pp. 73–77, Oct. 2012. doi: 10.1109/LDAV.2012.6378978 2

[19] K. Pöthkow, B. Weber, and H.-C. Hege. Probabilistic marching cubes. *Computer Graphics Forum*, 30(3):931–940, June 2011. doi: 10.1111/j.1467-8659.2011.01942.x 1, 2

[20] K. Potter, P. Rosen, and C. R. Johnson. From quantification to visualization: A taxonomy of uncertainty visualization approaches. In A. M. Dienstfrey and R. F. Boisvert, eds., *Uncertainty Quantification in Scientific Computing*, pp. 226–249. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. doi: 10.1007/978-3-642-32677-6_15 2

[21] J. Shen and H. Shen. Psrflow: Probabilistic super resolution with flow-based models for scientific data. *IEEE Transactions on Visualization and Computer Graphics*, 30(01):986–996, jan 2024. doi: 10.1109/TVCG.2023.3327171 2

[22] N. Shi, J. Xu, H. Li, H. Guo, J. Woodring, and H.-W. Shen. VDL-Surrogate: A view-dependent latent-based model for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):820–830, 2023. doi: 10.1109/TVCG.2022.3209413 1

[23] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020. 2

[24] Y. Sun and M. G. Genton. Functional boxplots. *Journal of Computational and Graphical Statistics*, 20(2):316–334, 2011. doi: 10.1198/jcgs.2011.09224 1

[25] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977. 1

[26] F. Vitart, C. Ardilouze, A. Bonet, A. Brookshaw, M. Chen, C. Codorean, M. Déqué, L. Ferranti, E. Fucile, M. Fuentes, et al. The Subseasonal to Seasonal (S2S) Prediction Project Database. *Bulletin of the American Meteorological Society*, 98(1), 2017. doi: 10.1175/BAMS-D-16-0017.1 2

[27] J. Wang, S. Hazarika, C. Li, and H.-W. Shen. Visualization and visual analysis of ensemble data: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2853–2872, 2019. doi: 10.1109/TVCG.2018.2853721 1

[28] Z. Wang, T. M. Athawale, K. Moreland, J. Chen, C. R. Johnson, and D. Pugmire. $FunMC^2$: A Filter for Uncertainty Visualization of Marching Cubes on Multi-Core Devices. In R. Bujack, D. Pugmire, and G. Reina, eds., *Eurographics Symposium on Parallel Graphics and Visualization*. The Eurographics Association, 2023. doi: 10.2312/pgv.20231081 2

[29] R. T. Whitaker, M. Mirzargar, and R. M. Kirby. Contour boxplots: A method for characterizing uncertainty in feature sets from simulation ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2713–2722, Dec. 2013. doi: 10.1109/TVCG.2013.143 1

[30] P. Zhan, G. Krokos, D. Guo, and I. Hoteit. Three-Dimensional Signature of the Red Sea Eddies and Eddy-Induced Transport. *Geophysical Research Letters*, 46(4), 2019. doi: 10.1029/2018GL081387 1, 2

[31] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020. 4