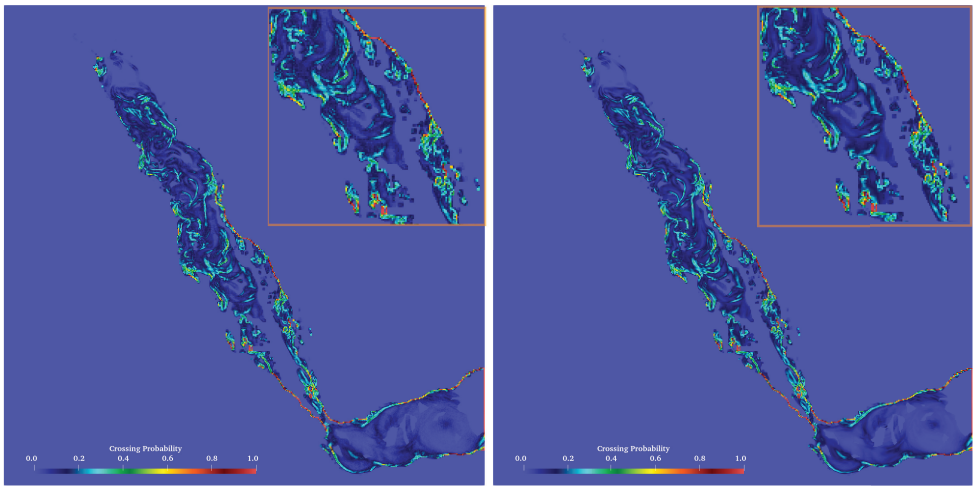
Accelerated Probabilistic Marching Cubes by Deep Learning for Time-Varying Scalar Ensembles

Mengjiao Han*
Scientific Computing and Imaging Institute

Tushar M. Athawale†
Oak Ridge National Laboratory

David Pugmire‡
Oak Ridge National Laboratory

Chris R. Johnson§
Scientific Computing and Imaging Institute



(a) Ground Truth (b) Predicted Result using Our Method

Figure 1: Visualizations of the level-crossing probability for isovalue 0.1 in the Red Sea data set [36] using our proposed neural network. Image (a) shows the level-crossing probabilities calculated using the original probabilistic marching cubes algorithm [25]. Image (b) shows the result computed by our trained model. The zoomed-in views are displayed in the top right. Our method can provide a visually identical result and is 10X faster than the original probabilistic marching cubes algorithm with parallel computing.

Abstract

Visualizing the uncertainty of ensemble simulations is challenging due to the large size and multivariate and temporal features of ensemble data sets. One popular approach to studying the uncertainty of ensembles is analyzing the positional uncertainty of the level sets. *Probabilistic marching cubes* is a technique that performs Monte Carlo sampling of multivariate Gaussian noise distributions for positional uncertainty visualization of level sets. However, the technique suffers from high computational time, making interactive visualization and analysis impossible to achieve. This paper introduces a deep-learning-based approach to learning the level-set uncertainty for two-dimensional ensemble data with a multivariate Gaussian noise assumption. We train the model using the first few time steps from time-varying ensemble data in our workflow. We demonstrate that our trained model accurately infers uncertainty in level sets for new time steps and is up to 170X faster than that of the original probabilistic model with serial computation and 10X faster than that of the original parallel computation.

Index Terms: Human-centered computing—Visualization—Visualization application domains—Scientific visualization; Computing methodologies—Machine learning—Machine learning approaches—Neural networks

1 Introduction

The increase in complexity and size of simulation data is often accompanied by uncertainties from multiple computational processes, such as simplifying assumptions and parameters of simulation models, noise from data reduction or interpolation, and errors in the mapping and rendering of visual attributes. Thus, uncertainty visualization continues to be one of the top research challenges for the visualization community [5, 19] since domain scientists can steer decision-making through reasoning based on uncertainty.

In recent years, an increasing number of tasks from scientific visualization, such as rendering [4, 17], data compression and reconstruction [8–11, 18, 22, 28], and feature extraction [21, 33, 35], have used deep-learning techniques, driving the performance to the next level. However, no research is currently employing neural networks for uncertainty visualization. In this paper, we investigate the feasibility of using deep learning-based methods for uncertainty visualization. We propose the first deep-learning-based method to address the challenge of positional uncertainty quantification of level sets in uncertain scalar fields for two-dimensional (2D) ensemble data sets.

We demonstrate that our method can learn uncertainties pertinent to underlying physics by evaluating the trained model with future data from the same simulation models. We compare the performance of our method to the probabilistic marching cubes, which was pioneered by Pöthkow et al. [25] to understand the spatial uncertainty of level sets. The probabilistic marching cubes technique has become popular in the past decade since it mitigates the occlusion and

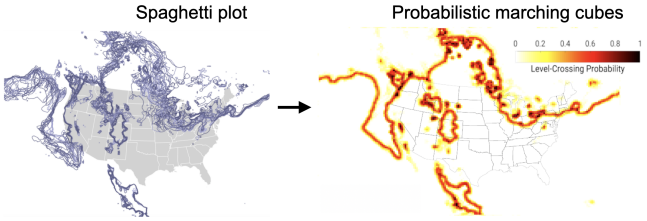


Figure 2: Spaghetti plot vs. probabilistic marching cubes for uncertainty visualization of level sets.

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

cluttering problems pertinent to spaghetti plots [26] of level sets. As illustrated in Fig. 2, the representation of spaghetti plots can be improved with the probabilistic marching cubes technique, which visualizes the most likely level-set positions in red and the uncertainty of level-set positions in yellow. Our proposed deep-learning method provides accuracy comparable to the original probabilistic model [25] (Fig. 1) for predictions of level-set probabilities. Our approach is up to 170X faster than the original probabilistic model with serial computation and up to 10X faster than the original model with parallel computation.

2 Related Work

2.1 Level-Set Uncertainty in Ensembles

Ensemble simulations are a popular way to capture uncertainty in simulations by performing simulations with different models and parameters. Analyzing uncertainty across ensemble members, however, can be challenging due to their large size and multivariate and temporal nature. Several researchers have investigated understanding uncertainty across ensembles of scientific simulations [6, 13, 26, 27, 31, 37]

Uncertainty of ensembles has been extensively studied via analyzing positional uncertainty of level-set visualizations [15, 34]. Pöthkow et al. [24, 25] proposed probabilistic models for visualization of positional uncertainty of level sets. Their approach comprised Monte Carlo sampling of multivariate Gaussian distributions [25] and nonparametric distributions [24] for uncertainty quantification. The Monte Carlo solutions, however, can be computationally challenging since computational time depends upon the number Monte Carlo samples. Athawale et al. provided closed-form accelerated solutions for uncertainty quantification of level sets [1–3] for independent parametric and nonparametric noise models. The closed-form solution, however, does not exist for the multivariate Gaussian noise assumption, thereby demanding expensive Monte Carlo sampling. Thus, we propose a deep-learning model for efficient computations of level-set uncertainty for the multivariate Gaussian noise assumption.

2.2 Deep Learning for Scientific Visualization

In recent years, deep-learning techniques have been increasingly studied for scientific visualizations [30, 32]. Their applications include interactive volume rendering [4, 17], parameter exploration for ensemble data [16], data super resolution [8–11, 18, 22], and feature extraction [21, 33, 35].

A convolutional neural network (CNN) is the most commonly used model architecture and is good for understanding spatial relations [21, 33, 35]. Combining CNN with generative adversarial network (GAN) [7] can produce high-resolution visualizations interactively [17] and provide more accurate super-resolution results [8, 10]. Moreover, some researchers employ multilayer perceptron (MLP) composed of a series of fully connected (FC) layers [22] to learn the implicit data representations.

However, research has not yet studied taking advantage of deep learning-based techniques for uncertainty visualization. In this paper, our goal is to employ deep-learning techniques to learn the positional uncertainty of isocontours for time-varying uncertain scalar fields, accelerating the probabilistic marching cubes algorithm [25].

3 Level-Crossing Probability Prediction using Deep Learning

We design our neural network to predict the level-crossing probability (LCP) of level sets for 2D time-varying ensemble data sets. The LCP is defined as the probability of a level set passing through a cell of an underlying grid [25]. Our technique comprises three steps. First, we generate the training data sets by quantifying the LCP through application of the original probabilistic marching cubes algorithm proposed by Pöthkow et al. [25] to the initial few time steps (Sect. 3.1). In the second step, the generated training samples are fed into a neural network built with MLP, which is adapted from the model architecture by Han et al. [12] and are applied sinusoidal activation functions inspired by Implicit Neural Representation [28] (Sect. 3.2). During the training process, the weights are optimized through backpropogation of the loss. Lastly, after the model is fully trained, the model is utilized for predicting the LCP for the rest of the time steps from the ensemble data (see supplemental material Sect. 2).

3.1 Training Data Generation

In our approach, we generate the training samples from the first t simulation time steps of the time-varying ensemble data and evaluate our method with the rest of the time steps. As per the probabilistic marching cubes algorithm [25], to compute the LCP for a given grid cell, the mean values and covariance matrix of the ensemble data are computed. Following the notation in [25], $Y = [Y_0, Y_1, Y_2, Y_3]$ represents random variables at the vertices of a 2D grid cell in one time step. Each vertex $Y_i = [y_i^0, y_i^1, ..., y_i^M]$ represents all members of the ensemble data, where $i = 0, 1, 2, 3$, and M is the number of ensemble members. We refer to $\hat{\mu} = [\hat{\mu_0}, \hat{\mu_1}, \hat{\mu_2}, \hat{\mu_3}]$ as the mean values of each grid vertex in a cell computed by averaging data across ensemble members. A 4×4 covariance matrix *Cov_Matrix* is computed per a 2D cell from the ensemble members to capture the pair-wise correlation between data at grid vertices i and j as:

$$\hat{Cov}_{i,j} = \frac{1}{M-1} \sum_{m=1}^{M} (y_i^m - \hat{\mu}_i)(y_j^m - \hat{\mu}_j) \quad (1)$$

where $i, j = 0, 1, 2, 3$, and M is the number of ensemble members.

To compute the LCP for an isovalue s in a 2D cell, r samples are randomly drawn from the multivariate Gaussian distribution of the means $\hat{\mu}$, and the covariance matrix *Cov_Matrix* is computed. If a level set passes through k samples, then the LCP p is computed as $p = k/r$ [25]. One training sample in our method represents one grid cell with a one-dimensional (1D) vector of size 16. The first four dimensions save the mean values ($\hat{\mu}$) of each grid point. Since the *Cov_Matrix* is symmetric, only the four variances (σ^2) on diagonal entries and the six covariances ($\hat{Cov}_{i,j}$) between the four grid points on nondiagonal entries are saved in the following 10 dimensions. Then the last two dimensions are the isovalue s and the corresponding LCP p (Equation 2). For our current study, we normalize data across all ensemble members before computing the LCP and select fixed isovalues of $[0.1, 0.2, ..., 0.9]$ that are all used in both training and testing for simplicity.

$$\begin{aligned} Training\_Sample = [&\hat{\mu_0}, \hat{\mu_1}, \hat{\mu_2}, \hat{\mu_3}, \sigma_0^2, \sigma_1^2, \sigma_2^2, \sigma_3^2, \\ &\hat{Cov}_{0,1}, \hat{Cov}_{0,2}, \hat{Cov}_{0,3}, \hat{Cov}_{1,2}, \hat{Cov}_{1,3}, \hat{Cov}_{2,3}, s, p] \end{aligned} \quad (2)$$

3.2 Network Architecture

We adapt the network architecture proposed by Han et al. [12] that consists of a latent encoder E and the latent decoder D, which are built with MLP, a series of FC layers (Figure 3). We refer to the vector of means as *vector_mean*, the vector of variances and covariances as *vector_cov*, and the vector of isovalues as *vector_iso*. These parameters are fed into three sequences of FC layers separately. The three outputs are concatenated into one latent vector and fed into another series of FC layers decoding to the LCP.

The sinusoidal activation function has been demonstrated to be more accurate and faster by Sitzmann et al. [28]. We adopted the sinusoidal activation function after each FC layer except the last layer of the latent decoder D. We applied the Sigmoid activation function before the output layer to guarantee the output is always between $[0, 1]$.

4 Results

We evaluated the network performance and compared our proposed deep learning-based approach with the original probabilistic marching cubes algorithm [25]. We demonstrated our proposed method

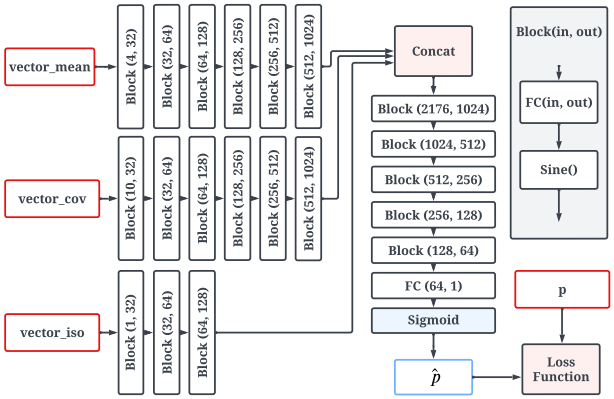


Figure 3: Network Architecture. The network of our method built with multilayer perceptrons (MLP) with the sinusoidal activation function. The network takes the mean values (*vector_mean*), variances and covariances (*vector_cov*), the iso-value (*vector_iso*), and the targeted LCP (p) as input, and outputs the predicted LCP ($\hat{p}$). The *vector_mean*, *vector_cov* and *vector_iso* are used for predicting the LCP $\hat{p}$ and the targeted LCP p is used to compute the loss.

is accurate and efficient by analyzing it on ensemble data sets from the IRI/LDEO Climate Data Library [1] and the Red Sea simulations performed at the Kaust Supercomputing Lab [2]. Models were trained on dual RTX 3090s GPUs and evaluated on a desktop equipped with an Intel(R) Xeon(R) W-3275M CPU (56 cores; 256GB memory) and one NVIDIA Titan RTX GPU.

4.1 Data Sets

Wind data set is from the European Center for Medium Range Weather Forecast (ECMWF) Sub-seasonal to Seasonal (S2S) Prediction Project [29]. The data set *pressure_level_wind* was obtained using the NECP ensemble forecast system with the *forecast* and *perturbed* parameters forming an ensemble with 15 members. We used *U_Component_Wind* ensemble with a pressure level at 200 HPA, a forecast hour at 0 on January 01, 2015, and a forecast period of 45 days. The spatial domain is from 0°E to 1.5°W in longitude and from 90°N to 90°S in latitude with a grid size of $[240 \times 121]$. In our experiments, the first 17 days' data were used to generate the training data sets, and we evaluated our study using the rest of the data. The total number of training samples is $776,306$.

Temperature data set is from the data set *sfc_temperature* at the same source as the **Wind** data set and with same forecast settings. The total number of training samples is $353,393$.

Red Sea [36] data set comprises ensemble simulations of variables relevant to the oceanology, such as velocity and temperature, performed over the domain with spatial resolution $500 \times 500 \times 50$ for 60 time steps. For our experiment, we analyzed the uncertainty in level sets of velocity magnitude across 10 ensemble members corresponding to the ocean surface (the top 2D data slice). We utilized ensembles for time steps $40-50$ for training the deep-learning model and used time steps $51-55$ as test data sets. The total number of training samples is $3,038,641$.

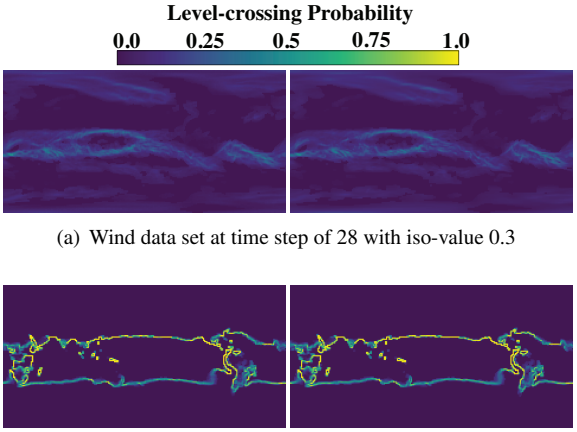
4.2 Network Performance

4.2.1 Training Performance

We trained the model with a fivefold cross-validation procedure and ran 100 epochs for each fold. The training time for the **Wind**, **Temperature** and **Red Sea** data is 1.5, 0.76, and 5.2 hours with dual RTX 3090s GPUs. The training time is linear to the number of training samples. The trained model requires a fixed storage space of 42M.

4.2.2 Visualization Performance

As shown in Fig. 1 and Fig. 4, the visualizations of the predicted LCP are indistinguishable from visualizations of the ground truth. Our proposed method performs surprisingly well for the **Red Sea** data set (Fig. 1), which has more noise compared to the **Wind** and **Temperature** data sets. In addition, by evaluating future time steps with similar underlying physics as the training data sets, our method can learn uncertainties relevant to the underlying physics.



(a) Wind data set at time step of 28 with iso-value 0.3

(b) Temperature data set at time step 22 with iso-value 0.8

Figure 4: Visualization of the inferred results (right column) using our trained neural network. The ground truth (left column) is computed using the probabilistic marching cubes algorithm with 8,000 Monte Carlo samples. Our trained model can predict the LCP visually indistinguishable from the ground truth.

Moreover, the quantitative analysis in Fig. 5 shows that our neural network can predict the LCP with a median error of less than 0.05 for all testing data sets.

We evaluated the impact of the number of training samples on the performance of our model quantitatively. We generated the entire data set by adding each time step in sequence and trained models using different percentages (from 10% to 100% with an increment of 10%) of the training samples from the entire training data set, which means we used the data in chronological order. Fig. 5 presents errors with an increase in the number of training samples, showing the results with five time steps for each data set. The errors are calculated pixel-wise between the predicted result and the ground truth. Fig. 5 reveals that there has been a steady decrease in the error with an increase in the number of training samples. However, the error reduction is less with more training samples.

4.2.3 Computational Performance and Comparisons

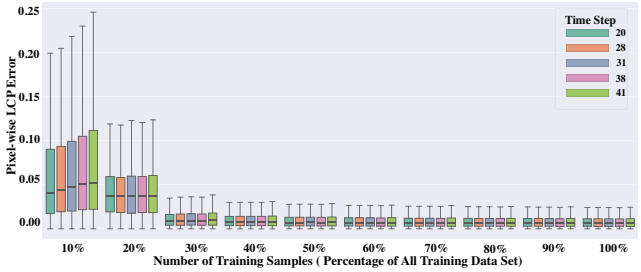
In Fig. 6, we compare the computation time using the probabilistic marching cubes algorithm with serial computation and parallel computation and compare them with the performance of our deep-learning method for LCP prediction. The first step for all methods is *Load Data*, including calculating means, variances, and covariances for each cell.

As discussed in [25], the vanilla probabilistic marching cubes algorithm is implemented in a serial computational fashion. The main weakness of this algorithm is the expensive computational time. To address this issue, we first parallelized the computation of the LCP over all grid cells. The parallelization is done by Joblib [3], a Python library for pipelining tasks. Figures on the left in Fig. 6 present the running time for serial and parallel computation using the probabilistic marching cubes algorithm. From these figures, we

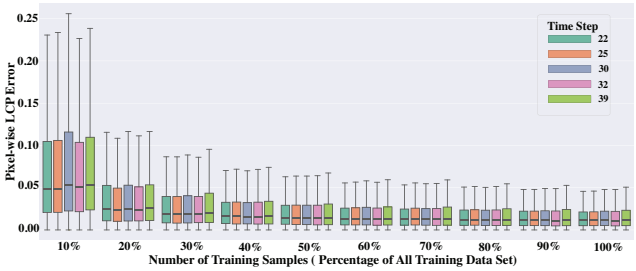
[1] http://iridl.ldeo.columbia.edu/

[2] https://kaust-vislab.github.io/SciVis2020/

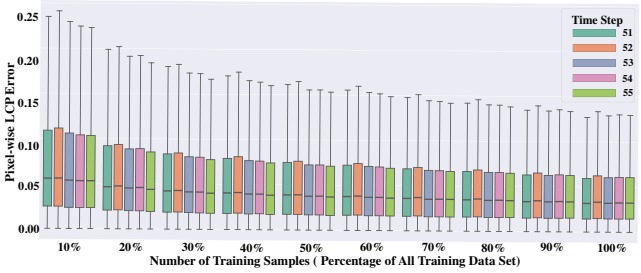
[3] https://joblib.readthedocs.io/en/latest/



(a) Wind data set



(b) Temperature data set



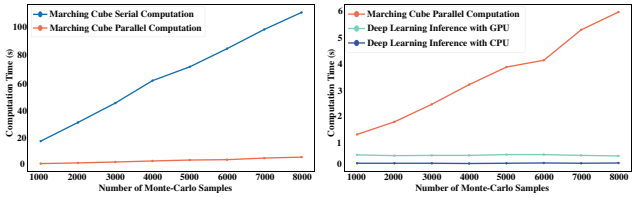
(c) Red Sea data set

Figure 5: Boxplots visualizing errors for the model trained using an increasing number of training samples. The errors are calculated as the pixel-wise absolute differences between model predicted results and the ground truth. The evaluations are performed using the test data from the five time for each data set. The outlier error values are not displayed in the boxplots. The results show that the inference accuracy can be improved by increasing the number of training samples, but the improvement is less with more training samples.

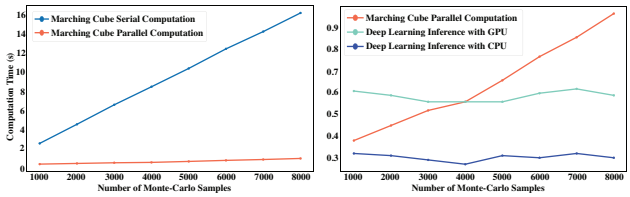
can conclude that the parallel version is at least 15X and up to 19X faster than the serial version across all tested data.

To understand and compare the performance of our neural network, we ran evaluations with the same testing data sets using the trained model with GPU and CPU. By comparing the computational time to the parallel version of probabilistic marching cubes, as shown in Fig. 6 (right), our deep-learning method can provide further speed-up of up to 10X faster (which amounts to an average speed-up of 170X compared to the serial version). In addition, we observed that the inference using CPU is slightly faster than using GPU with the trained neural network. A possible explanation for this might be that the predicted results need to be transferred back to the CPU for other processes, such as saving as a file or plotting visualizations, when using GPU for inferences, whereas using the CPU avoids this transferring process.

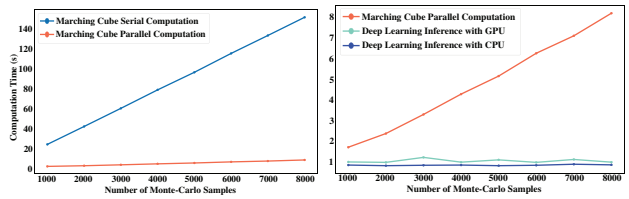
Moreover, our deep learning method requires loading the trained model before computation. Our models cost about 1.7s to load. This process needs to be performed only once, and the loaded model can be employed for multiple computations.



(a) Wind data set. Evaluated using time step of 33 with isovalue of 0.2.



(b) Temperature data set. Evaluated using time step of 22 with isovalue of 0.8.



(c) Red Sea data set. Evaluated using time step of 54 with isovalue of 0.1.

Figure 6: The computational time over the number of Monte Carlo samples. In the left column, we compare the performance of the serial probabilistic marching cubes algorithm [25] with a parallel version. The right column depicts the performance of our deep-learning method with CPU and GPU inferences. The parallel probabilistic marching cubes algorithm is up to 19X faster than the serial version, and our deep learning method further accelerates the parallel probabilistic marching cubes approach up to 10X.

5 Conclusion and Future Work

We propose a deep neural network to predict the positional uncertainty of level sets for uncertain time-varying scalar ensemble data. We have contributed the first assessment of applying deep learning to uncertainty visualization. Our study demonstrates that our model used to accurately predict level-crossing probabilities for 2D ensemble data sets and can learn the uncertainties relevant to the underlying physics. More importantly, our method is up to 170X faster than the original probabilistic marching cubes technique with serial computations and up to 10X faster compared to the parallel version of probabilistic marching cubes according to our experiments. In the future, we plan to extend our method to three-dimensional data. Besides being limited to a fixed set of isovalues, we would like to enhance the flexibility of predicting for varying isovalues. Further, we would like to investigate the applicability of our approach in the context of other visualization techniques, such as volume rendering [20] and flow visualizations [23], that utilize the Monte Carlo approach for uncertainty quantification. We would also like to expand our work to Monte Carlo techniques that utilize more sophisticated probability distribution models, e.g., copula-based distributions [14], for uncertainty quantification.

Acknowledgments

This work was partially supported by the Intel Graphics and Visualization Institutes of XeLLENCE, the Intel OneAPI CoE, the NIH under award R24 GM136986, the DOE under grant number DE-FE0031880, the Utah Office of Energy Development, and Scientific Discovery through Advanced Computing (SciDAC) program in U.S. Department of Energy.

REFERENCES

[1] T. M. Athawale and A. Entezari. Uncertainty Quantification in Linear Interpolation for Isosurface Extraction. *IEEE Transactions on Visualization and Computer Graphics*, 19(12), 2013. doi: 10.1109/TVCG.2013.208

[2] T. M. Athawale, E. Sakhaee, and A. Entezari. Isosurface Visualization of Data with Nonparametric Models for Uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 22(1), 2016. doi: 10.1109/TVCG.2015.2467958

[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)*, 2021. doi: 10.1109/VIS49827.2021.9623267

[4] M. Berger, J. Li, and J. A. Levine. A Generative Model for Volume Rendering. *IEEE transactions on visualization and computer graphics*, 25(4), 2018. doi: 10.1109/TVCG.2018.2816059

[5] K. Brodlie, R. A. Osorio, and A. Lopes. A Review of Uncertainty in Data Visualization. In *Expanding the Frontiers of Visual Analytics and Visualization*. Springer Verlag London, 2012. doi: 10.1007/978-1-4471-2804-5_6

[6] H. Chen, S. Zhang, W. Chen, H. Mei, J. Zhang, A. Mercer, R. Liang, and H. Qu. Uncertainty-Aware Multidimensional Ensemble Data Visualization and Exploration. *IEEE Transactions on Visualization and Computer Graphics*, 21(9), 2015. doi: 10.1109/TVCG.2015.2410278

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. *Advances in neural information processing systems*, 27, 2014.

[8] L. Guo, S. Ye, J. Han, H. Zheng, H. Gao, D. Z. Chen, J.-X. Wang, and C. Wang. SSR-VFD: Spatial Super-Resolution for Vector Field Data Analysis and Visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, 2020. doi: 10.1109/PacificVis48177.2020.8737

[9] 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), 2019. doi: 10.1109/TVCG.2019.2934255

[10] J. Han and C. Wang. SSR-TVD: Spatial Super-Resolution for Time-Varying Data Analysis and Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2020. doi: 10.1109/TVCG.2020.3032123

[11] J. Han, H. Zheng, D. Z. Chen, and C. Wang. STNet: An End-to-End Generative Framework for Synthesizing Spatiotemporal Super-Resolution Volumes. *IEEE Transactions on Visualization and Computer Graphics*, 28(1), 2021. doi: 10.1109/TVCG.2021.3114815

[12] M. Han, S. Sane, and C. R. Johnson. Exploratory Lagrangian-Based Particle Tracing Using Deep Learning. *Journal of Flow Visualization and Image Processing*, 2022. doi: 10.1615/JFlowVisImageProc.2022041197

[13] L. Hao, C. G. Healey, and S. A. Bass. Effective Visualization of Temporal Ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 22(1), 2016. doi: 10.1109/TVCG.2015.2468093

[14] S. Hazarika, A. Biswas, and H.-W. Shen. Uncertainty Visualization Using Copula-Based Analysis in Mixed Distribution Models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):934–943, 2018. doi: 10.1109/TVCG.2017.2744099

[15] S. Hazarika, S. Dutta, and H.-W. Shen. Visualizing the Variations of Ensemble of Isosurfaces. In *2016 IEEE Pacific Visualization Symposium (PacificVis)*, 2016. doi: 10.1109/PACIFICVIS.2016.7465272

[16] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. Nashed, and T. Peterka. InSituNet: Deep Image Synthesis for Parameter Space Exploration of Ensemble Simulations. *IEEE transactions on visualization and computer graphics*, 26(1), 2019. doi: 10.1109/TVCG.2019.2934312

[17] F. Hong, C. Liu, and X. Yuan. DNN-VolVis: Interactive Volume Visualization Supported by Deep Neural Network. In *2019 IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 2019. doi: 10.1109/PacificVis.2019.00041

[18] J. Jakob, M. Gross, and T. Günther. A Fluid Flow Data Set for Machine Learning and its Application to Neural Flow Map Interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 27(2), 2020. doi: 10.1109/TVCG.2020.3028947

[19] C. R. Johnson. Top Scientific Visualization Research Problems. *IEEE Computer Graphics and Applications*, 24(4), 2004. doi: 10.1109/MCG.2004.20

[20] 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, 2012. doi: 10.1109/LDAV.2012.6378978

[21] Y. Liu, Y. Lu, Y. Wang, D. Sun, L. Deng, F. Wang, and Y. Lei. A CNN-based shock detection method in flow visualization. *Computers & Fluids*, 184, 2019. doi: 10.1016/j.compfluid.2019.03.022

[22] Y. Lu, K. Jiang, J. A. Levine, and M. Berger. Compressive Neural Representations of Volumetric Scalar Fields. In *Computer Graphics Forum*, vol. 40. Wiley Online Library, 2021. doi: 10.1111/cgf.14295

[23] M. Otto, T. Germer, H.-C. Hege, and H. Theisel. Uncertain 2D Vector Field Topology. *Computer Graphics Forum*, 29(2):347–356, 2010.

[24] K. Pöthkow and H.-C. Hege. Nonparametric Models for Uncertainty Visualization. *Computer Graphics Forum*, 32(3pt2), 2013. doi: 10.1111/cgf.12100

[25] K. Pöthkow, B. Weber, and H.-C. Hege. Probabilistic Marching Cubes. In *Computer Graphics Forum*, vol. 30. Wiley Online Library, 2011. doi: 10.1111/j.1467-8659.2011.01942.x

[26] K. Potter, A. Wilson, P.-T. Bremer, D. Williams, C. Doutriaux, V. Pascucci, and C. R. Johnson. Ensemble-Vis: A Framework for the Statistical Visualization of Ensemble Data. In *2009 IEEE International Conference on Data Mining Workshops*, 2009. doi: 10.1109/ICDMW.2009.55

[27] J. Sanyal, S. Zhang, J. Dyer, A. Mercer, P. Amburn, and R. Moorhead. Noodles: A Tool for Visualization of Numerical Weather Model Ensemble Uncertainty. *IEEE Transactions on Visualization and Computer Graphics*, 16(6), 2010. doi: 10.1109/TVCG.2010.181

[28] 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, 2020. doi: 10.48550/arXiv.2006.09661

[29] 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

[30] C. Wang and J. Han. DL4SciVis: A State-of-the-Art Survey on Deep Learning for Scientific Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2022. doi: 10.1109/TVCG.2022.3167896

[31] 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), 2019. doi: 10.1109/TVCG.2018.2853721

[32] Q. Wang, Z. Chen, Y. Wang, and H. Qu. A Survey on ML4VIS: Applying Machine Learning Advances to Data Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2021. doi: 10.1109/TVCG.2021.3106142

[33] Y. Wang, L. Deng, Z. Yang, D. Zhao, and F. Wang. A rapid vortex identification method using fully convolutional segmentation network. *The Visual Computer*, 37(2), 2021. doi: 10.1007/s00371-020-01797-6

[34] 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), 2013. doi: 10.1109/TVCG.2013.143

[35] T. B. L. Yi. CNN-based Flow Field Feature Visualization Method. *International Journal of Performability Engineering*, 14(3), 2018. doi: 0.23940/ijpe.18.03.p4.434444

[36] 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

[37] M. Zhang, L. Chen, Q. Li, X. Yuan, and J. Yong. Uncertainty-Oriented Ensemble Data Visualization and Exploration using Variable Spatial Spreading. *IEEE Transactions on Visualization and Computer Graphics*, 27(2), 2021. doi: 10.1109/TVCG.2020.3030377