GNN-Surrogate: A Hierarchical and Adaptive Graph Neural Network for Parameter Space Exploration of Unstructured-Mesh Ocean Simulations

Neng Shi, Jiayi Xu, Skylar W. Wurster, Hanqi Guo, *Member, IEEE,* Jonathan Woodring, Luke P. Van Roekel, and Han-Wei Shen, *Member, IEEE*

1 Additional Related Work: Deep Generative Models for Scientific Visualization

Deep generative models (e.g., PixelRNN [1], PixelCNN [2], VAE [3], and GAN [4], etc.) have been successfully used in various applications, including image generation [5], [6], image-to-image translation [7], [8], text-to-image translation [9], [10], super resolution [11], and 3D object generation [12], [13]. Scientific visualization researchers increasingly use deep generative models to synthesize visualization results [14]–[18]. As opposed to directly generating visualization results, other works focus on predicting 3D fields. Super resolution [19]–[23], simulation result prediction [24], [25], variable selection and translation [26], and geometric model prediction [27] are some examples. Our work also focuses on predicting 3D fields. However, our fields are represented with unstructured meshes, making it hard to apply regular neural networks such as CNNs unless we first resample the data onto 3D regular grids, which would lead to inevitable oversampling or undersampling in certain areas.

2 Resampling onto Regular Grids

One way of processing unstructured grids is to resample the simulation data onto regular grids. In this section, we experimented using the resampling method and showed the results. For each MPAS-Ocean horizontal spherical layer, we resample the temperature field onto Cartesian girds with a similar resolution to the Voronoi tessellations, i.e., we represent the temperature field with rectilinear grids.

- *Neng Shi, Jiayi Xu, Skylar W. Wurster and Han-Wei Shen are with the Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, 43210, USA.*
  E-mail: {shi.1337, xu.2205, wurster.18, shen.94}@osu.edu
- *Hanqi Guo is with the Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439, USA.*
  E-mail: hguo@anl.gov
- *Jonathan Woodring is with the Applied Computer Science Group (CCS-7), Los Alamos National Laboratory, Los Alamos, NM 87544.*
  Email: woodring@lanl.gov
- *Luke P. Van Roekel is with the Fluid Dynamics and Solid Mechanics Group (T-3), Los Alamos National Laboratory, Los Alamos, NM 87544.*
  Email: lvanroekel@lanl.gov

TABLE 1
Quantitative comparison of GNN-Surrogate and the resampling method.

	GNN-Surrogate	resampling
PSNR (global, dB)	**50.7**	25.1
MD (global)	**0.1965**	0.9975
PSNR (ROI, dB)	**39.5**	29.3
MD (ROI)	**0.1774**	1.428

TABLE 2
The Graph Hierarchy. "Operation" shows the coarsening operations we use from previous to the current level. "H" and "V" indicates horizontal and vertical edge matching.

Level	Operations	Nodes
0	-	11845146
3	HHV	1156304
6	HHV	207500
8	HH	55536
11	HHV	7990
14	HHV	1170
16	HH	341
18	HH	105

We resampled from the rectilinear grid back to the Voronoi tesselation and compared with the ground truth, and give the data-level quantitative result of the resampling method in Table 1, which is much worse than using graph convolutions. The uniform resolution makes fine details at some locations lost, leading to bad results.

3 Graph Coarsening Details

This section shows the details for graph coarsening. Starting from the original graph G_0, we alternately perform horizontal and vertical edge matching to generate graphs at different resolution levels $G_1, G_2, \ldots, G_{L-1}$. Only listed levels in Table 2 are used for GNN-Surrogate design.

4 Graph Transformation Algorithm

In the primary text, given a remaining graph hierarchical tree after cutting C^*, we transform the original

Algorithm 1 Transform the graphs $\{G_0, G_1, \ldots, G_{L-1}\}$ given the remaining graph hierarchical tree (GHT) after cutting C^*.

```
Input:
    The graph hierarchy $\{G_0, G_1, \ldots, G_{L-1}\}$;
    The edge attribute vectors $\{\mathbf{\Gamma_0}, \mathbf{\Gamma_1}, \ldots, \mathbf{\Gamma_{L-1}}\}$;
    The remaining GHT after cutting $C^*$;
    The direction set $\mathcal{D}$;
Output:
    Transformed graph hierarchy $\{G'_0, G'_1, \ldots, G'_{L-1}\}$;
    The edge attribute vectors for transformed graphs $\{\mathbf{\Gamma'_0}, \mathbf{\Gamma'_1}, \ldots, \mathbf{\Gamma'_{L-1}}\}$;
 1: for each $l \in [0, L-1]$ do
 2:    for each $v_{l,j} \in V_l$ do
 3:       Add the proxy node $\psi_{C^*}(v_{l,j})$ into $V'_l$;
 4:    end for
 5:    for each $p_A \in V'_l$ do
 6:       for each $p_B \in V'_l$ do
 7:          Set $E_{AB} = \{(v_{l,i}, v_{l,j}) \mid \psi_{C^*}(v_{l,i}) = p_A, \psi_{C^*}(v_{l,j}) = p_B\}$;
 8:          if $p_A \neq p_B$ and $|E_{AB}| > 0$ then
 9:             Add $(p_A, p_B)$ into $E'_l$
10:             $\mathbf{\Gamma'_l}(p_A, p_B) = \frac{\sum_{(v_{l,i}, v_{l,j}) \in E_{AB}} \mathbf{\Gamma_l}(v_{l,i}, v_{l,j})}{|E_{AB}|}$;
11:          end if
12:       end for
13:    end for
14:    Set $G'_l = (V'_l, E'_l)$
15: end for
16: return $\{G'_0, G'_1, \ldots, G'_{L-1}\}, \{\mathbf{\Gamma'_0}, \mathbf{\Gamma'_1}, \ldots, \mathbf{\Gamma'_{L-1}}\}$
```

graph hierarchy $\{G_0, G_1, \ldots, G_{L-1}\}$ with an edge attribute vector list $\{\mathbf{\Gamma_0}, \mathbf{\Gamma_1}, \ldots, \mathbf{\Gamma_{L-1}}\}$ to a transformed graph hierarchy $\{G'_0, G'_1, \ldots, G'_{L-1}\}$ with an edge attribute vector list $\{\mathbf{\Gamma'_0}, \mathbf{\Gamma'_1}, \ldots, \mathbf{\Gamma'_{L-1}}\}$. Algorithm 1 shows the graph transformation pseudocode.

5 Adversarial Training

Adversarial training can help generate local high-frequency features [17], [20], [21], [21], [26]. In this section, we explain the exatra network architecture of GNN-Surrogate and related operations for adversarial training. We show experiments in Section 7 to see whether adversarial training helps us obtain an surrogate model with higher quality.

5.1 Architecture

During adverarial training, two subnetworks are involved in GNN-Surrogate: a generator R and a discriminator D. The upsampling-convolution generator R is trained to generate outputs that can not be distinguished from "real" adaptive resolution residuals. On the other hand, the convolution-pooling discriminator D is trained to try its best to tell the "fake" output of R from actual residuals. The generator is the same as without adversarial training. We introduce the discriminator D below.

Figure 1 shows the architecture of D. A discriminator is used to define the adversarial loss and can capture high-frequency features, which is demonstrated in previous work [7], [14], [17]. D takes adaptive resolution residual data as input and outputs a scalar value indicating the input's realness. A forward pass in the discriminator

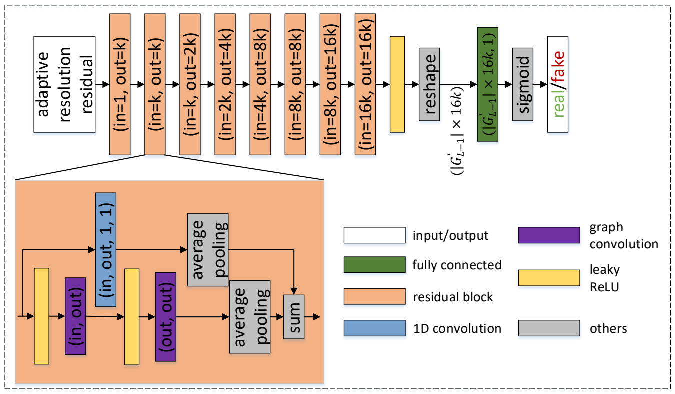

Fig. 1. The architecture of discriminator D.

contains two steps. First, the input goes through some residual blocks so that we obtain a latent representation. The residual block shares a similar architecture with the one in R except that pooling replaces upsampling and there is no instance normalization. Second, we feed the latent representation to a fully connected layer for the realness value. Motivated by Alec et al. [28], we use Leaky ReLU [29] as the activation function instead of ReLU in all layers except for the last layer. In the last layer for binary classification, *sigmoid* is used to output a scalar value, ensuring the range is within $[0, 1]$.

5.2 Graph Pooling

We pool a graph of level l to a graph of level m, where $m > l$. The value of a node at level m equals the average of descendants at level l. Given a feature map $\Omega_l \in \mathbb{R}^{|V'_l| \times c}$ on graph G'_l, the graph pooling outputs $\Upsilon_m \in \mathbb{R}^{|V'_m| \times c}$ on graph G'_m, such that

$$\Upsilon_{m,i} = \frac{\sum_{v_{l,j} \in \tau(m,l,i)} \Omega_{l,j}}{|\tau(m,l,i)|}, \tag{1}$$

where $\tau(m,l,i) = \{v_{l,j} \mid v_{m,i} \in V'_m$ is the ancestor of $v_{l,j} \in V'_l$ in the GHT$\}$ is node $v_{m,i}$'s descendant nodes in V'_l.

5.3 Training Process

In the training process, following the standard approach from Goodfellow et al. [4], we update the discriminator and generator alternatively using gradient descent. We also explain other used methods during training as follows.

Loss Function During training, we iteratively update parameters in GNN-Surrogate to minimize a loss function. Our loss function is a mixture of a traditional point-wise loss, i.e., the L_1 loss, and an adversarial loss L_{adv_R}. The loss can guide R to produce results with real global structure and high-frequency features, as mentioned in some previous work [7]. For R, the loss function is

$$\begin{aligned} L &= L_1 + \lambda L_{adv\_R}, \\ L_1 &= \frac{1}{b} \textstyle\sum_{i=0}^{b-1} ||\hat{S}_{ar,i} - S_{ar,i}||_1, \\ L_{adv\_R} &= -\frac{1}{b} \textstyle\sum_{i=0}^{b-1} logD(\hat{S}_{ar,i}), \end{aligned} \tag{2}$$

where b is the batch size, $S_{ar,0:b-1}$ and $\hat{S}_{ar,0:b-1}$ are ground truth and generated adaptive resolution residual, λ is the coefficient used to balance two loss terms. The loss function of the discriminator D is

$$L_D = -\frac{1}{b} \sum_{i=0}^{b-1} (logD(S_{ar,i})) + log(1 - D(\hat{S}_{ar,i})). \tag{3}$$

Training Techniques Besides spectral normalization [30] and mixed precision [31] mentioned in the primary text, we apply the two time-scale update rule (TTUR) [32] to stabilize the adversarial training. The details of our training techniques can be found in Section 6.2.

6 Network Training Details

6.1 Training Configuration

During GNN-Surrogate training, we set the batch size to 1 since training the model is still costly after applying mixed precision training. We use Adam [33] as the optimizer, with $\beta_1 = 0.0$, $\beta_2 = 0.999$. We set the learning rate of the generator $\alpha_R = 5 \times 10^{-5}$. During adversarial training, we follow TTUR and set our discriminator's learning rate to 4 times that of the generator, i.e., $\alpha_D = 2 \times 10^{-4}$.

6.2 Training Techniques Details

6.2.1 *Spectral Normalization*

In the adversarial training process, we want to constrain the discriminator because, during training, when the support of the model distribution and target distribution (real data) do not intersect, there is a discriminator that can easily distinguish the model distribution from the target distribution. In this case, the discriminator has a gradient of 0 with respect to the input, so the update of the generator will stop. Spectral normalization [30] is a weight normalization technique that avoids this situation. Recent work [34] shows that applying spectral normalization to generators also helps since it can avoid mode collapse and prevent unusual gradients. We apply spectral normalization to all the fully connected and graph convolution layers in both the generator R and the discriminator D (if exists).

6.2.2 *Two Time-scale Update Rule (TTUR)*

During adversarial training, when optimizing R, we assume that D's discriminating power is better than R's generating ability so that D can guide R to learn in a better direction. The usual approach is to update D's parameters more than once and then update R's parameters once, which is computationally expensive. TTUR [32] is a simpler update strategy, which is to set different learning rates for D and R to help D converge faster. We utilize this technique in GNN-Surrogate adversarial training.

6.2.3 *Mixed Precision Training*

Training a deep generative model for large-scale data is memory costly and computationally expensive. We counter the hardware limitations by using mixed precision training [31]. Modern deep learning systems use a single-precision (FP32) format. We use IEEE half-precision format (FP16) to train all the layers except for Instance Normalization because of numerical stability. Networks can match the accuracy compared with the one training with FP32, and we do not need to modify the model or adjust the hyperparameters. The technique enables us to reduce the GPU memory cost by half and drastically speed up the training.

7 Hyperparameter Studies

Choosing hyperparameters for GNN-Surrogate is nontrivial. In this section, we evaluated the influence of different hyperparameters (i.e., number of simulation runs needed for the cutting policy, loss functions, the threshold used for the GHT cuting, and network architectures). We also evaluated the influence of different number of sampled data instances for the baseline method inverse distance weighting interpolation.

TABLE 3
The Graph Hierarchical Tree cutting result ($\varepsilon = 0.5$). Different ensemble members are used for the cutting.

Level	Nodes (1)	Nodes (16)	Nodes (32)	Nodes (70)
0	1069790	1794330	1816813	1834123
3	231491	345284	348028	350950
6	49240	66561	66871	67308
8	17631	22467	22516	22631
11	3675	4317	4317	4328
14	740	811	811	814
16	256	274	274	274
18	98	101	101	101

TABLE 4
Quantitative evaluation of GNN-Surrogate trained with different loss functions. The model trained with L_1 generates results with the best PSNR and MD at the data-level compared with the model trained with the combination of L_1 and L_{R_adv}.

	L_1	$L_1 + 10^{-3}L_{R\_adv}$	$L_1 + 10^{-2}L_{R\_adv}$
PSNR (global, dB)	**50.7**	49.3	49.9
MD (global)	**0.1965**	0.2186	0.2229
PSNR (ROI, dB)	**39.5**	38.1	39.3
MD (ROI)	**0.1774**	0.1932	0.1952

7.1 Number of Simulation Runs Needed for Cutting Policy

In the primary text, we state that 16 simulation outputs were used for the graph hierarchical tree cutting generation. This section gives a detailed analysis of how we chose 16 as the number of simulation outputs. In the training dataset containing 70 simulation outputs, we select 1, 16, 32, and 70 outputs to generate the tree cutting. Table 3 shows each level's node numbers in the remaining tree using different numbers of simulation outputs. We can see that the cutting results are similar when we use more than 16 simulation outputs. Thus, we chose 16 as the number of simulation outputs.

7.2 Loss Functions

We evaluated GNN-Surrogate trained with the traditional point-wise loss L_1, and the combination of L_1 and L_{R_adv} using different coefficient settings $\lambda(10^{-2}, 10^{-3})$.

Table 4 reports the quantitative results at the data-level using different loss functions. We found that using L_1 gave us the best PSNR and MD. Furthermore, Figure 2 shows the the quantitative results at the geometry-level and image-level. We did not find that the results from the model trained with a combination of L_1 and L_{R_adv} are obviously better in SSIM and EMD than the results of the model trained with L_1 only. Therefore, we decided to use L_1 as our default loss function for GNN-Surrogate training.

7.3 Threshold Used for Cutting

As mentioned in the primary text, the threshold ε used for the GHT cut controls the size of transformed graphs and adaptive resolution temperature (ART) fields. In this experiment, we evaluated three ε values: 0.25, 0.5, and 0.75.

Table 8 shows the quantitative evaluation results, where we have two findings. First, all three ε values lead to good predicted

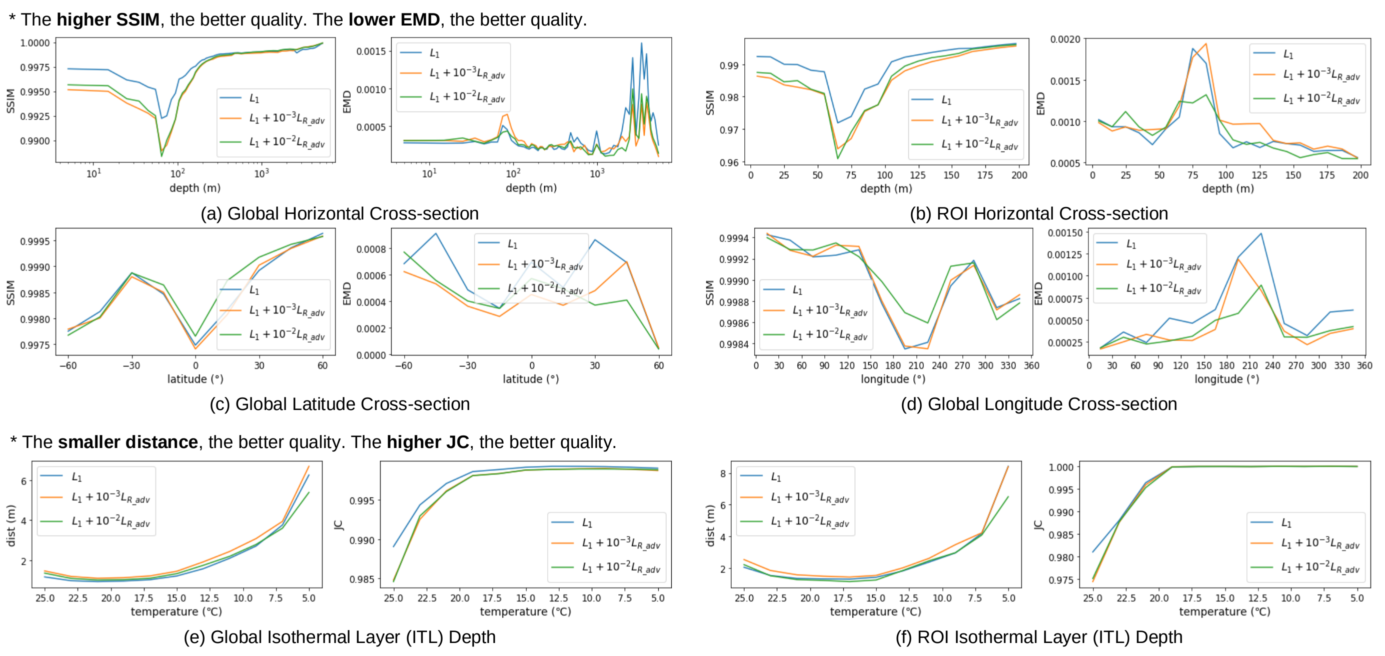

Fig. 2. Quantitative evaluation of GNN-Surrogate trained with different loss functions. (a-b) SSIM and EMD for temperature horizontal cross-sections at different depths. (c) SSIM and EMD for vertical cross-sections at different latitudes. (d) SSIM and EMD for vertical cross-sections at different longitudes. (e-f) Average distance and Jaccard Coefficient (JC) for the isothermal layer (ITL) depth maps with different temperature isovalues.

TABLE 5
Quantitative evaluation of different GHT cut threshold.

# threshold ε	0.25	0.50	0.75
PSNR (Global, dB)	**51.0**	50.7	49.8
MD (Global)	0.2127	0.1965	**0.1898**
PSNR (ROI, dB)	**40.2**	39.5	38.4
MD (ROI)	0.2037	**0.1774**	0.1952
Graph Size (MB)	1034	661	**494**
Model Size (MB)	**218**	545	815
ART Size (MB)	24.27	13.69	**9.09**

results' accuracy. Second, a smaller ε leads to larger simplified graphs and ART fields. A larger ε causes a larger model size since we adaptively increase the size of GNN-Surrogate to fully occupy the 16GB GPU memory. The 0.5 ε balances the size of transformed graphs, ART fields, and models well. Thus, the threshold for the GHT cut ε is set to 0.5.

7.4 Network Architectures

The channel multiplier k controls our network's architecture, and different network architectures require different computational power. Under the condition that the GHT cut threshold ε and the loss function used for training are set, we evaluated four k values: 16, 32, 64, 96.

Table 6 shows the data-level quantitative evaluation results. The training time was fixed to 36 hours. We can see that in general, larger k leads to a more powerful GNN-Surrogate model and more accurate predicted results. However, decreasing k from 96 to 32 does not cause the results to worsen much. This is a good news since we are able to reduce the GPU memory during training and the disk space for model snapshots. When $k = 32$, GNN-Surrogate training only costs 7.88GB GPU memory, which means theoretically, the training can be achieved on a NVIDIA GEFORCE GTX 1080 Ti.

TABLE 6
Quantitative evaluation of different network architectures controlled by k.

# k	16	32	64	96
PSNR (global)	48.3	49.9	49.8	**50.7**
MD (global)	0.3591	0.2140	0.3914	**0.1965**
PSNR (ROI)	38.5	39.5	**40.3**	39.5
MD (ROI)	0.2083	0.1850	0.1873	**0.1774**
Model Size (MB)	**59**	130	311	545
GPU cost (GB)	**6.00**	7.88	11.56	15.38

TABLE 7
Quantitative evaluation of the inverse distance weighting interpolation with different number of sampled data instances g.

# g	1	2	3	4	5
PSNR (global)	46.9	46.6	**47.7**	47.2	47.3
MD (global)	0.2035	0.1863	0.1721	0.1756	**0.1678**
PSNR (ROI)	32.7	32.4	**33.6**	33.1	33.1
MD (ROI)	0.2000	0.1838	**0.1673**	0.1740	0.1712

7.5 Number of Sampled Data Instances for Inverse Distance Weighting Interpolation

As mentioned in the primary text, for the baseline method inverse distance weighting interpolation, we sampled g data instances whose parameter settings have the minimum Manhattan distance to the test data, and applied the weighted sum. In this experiment, Values of g from 1 to 5 were evaluated.

Table 7 reports the quantitative results at the data-level. Figure 3 reports the quantitative results at the geometry-level and image-level. We find that setting $g = 3$ balances the metrics in three levels, thus we present the results for $g = 3$ in the primary text.

8 Summary of Mathematical Symbols

Table 8 is a summary of mathematical symbols.

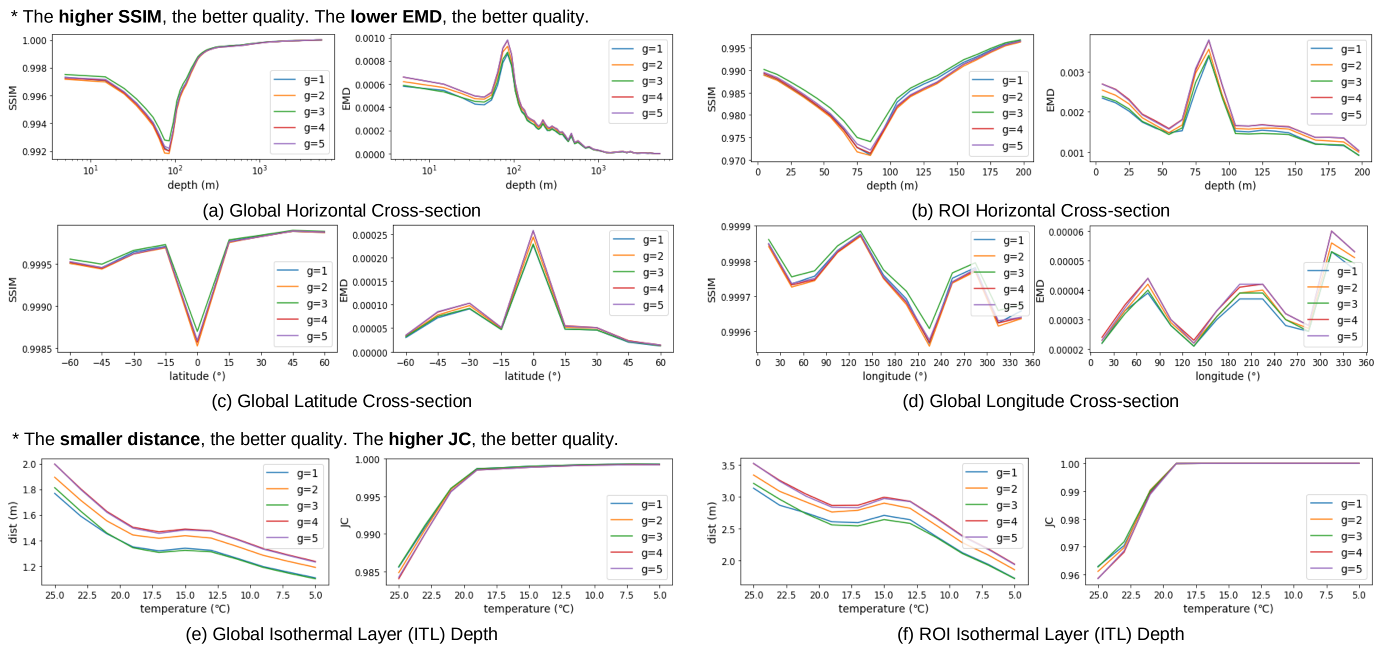

Fig. 3. Quantitative evaluation of the inverse distance weighting interpolation with different number of sampled data instances g. (a-b) SSIM and EMD for temperature horizontal cross-sections at different depths. (c) SSIM and EMD for vertical cross-sections at different latitudes. (d) SSIM and EMD for vertical cross-sections at different longitudes. (e-f) Average distance and Jaccard Coefficient (JC) for the isothermal layer (ITL) depth maps with different temperature isovalues.

References

[1] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks," *arXiv preprint arXiv:1601.06759*, 2016.

[2] A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves *et al.*, "Conditional Image Generation with PixelCNN Decoders," in *Advances in Neural Information Processing Systems*, 2016, pp. 4790–4798.

[3] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[5] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive Growing of GANs for Improved Quality, Stability, and Variation," in *Proc. International Conference on Learning Representations*, 2018.

[6] Y. Jin, J. Zhang, M. Li, Y. Tian, H. Zhu, and Z. Fang, "Towards the Automatic Anime Characters Creation with Generative Adversarial Networks," *arXiv preprint arXiv:1708.05509*, 2017.

[7] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image Translation with Conditional Adversarial Networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1125–1134.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.

[9] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "StackGAN: Text to Photo-Realistic Image Synthesis with Stacked Generative Adversarial Networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5907–5915.

[10] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative Adversarial Text to Image Synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[11] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4681–4690.

[12] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling," in *Advances in Neural Information Processing Systems*, 2016, pp. 82–90.

[13] M. Gadelha, S. Maji, and R. Wang, "3D Shape Induction from 2D Views of Multiple Objects," in *Proc. International Conference on 3D Vision (3DV)*. IEEE, 2017, pp. 402–411.

[14] M. Berger, J. Li, and J. A. Levine, "A Generative Model for Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 4, pp. 1636–1650, 2018.

[15] F. Hong, C. Liu, and X. Yuan, "DNN-VolVis: Interactive Volume Visualization Supported by Deep Neural Network," in *IEEE Pacific Visualization Symposium (PacificVis)*. IEEE, 2019, pp. 282–291.

[16] D. Engel and T. Ropinski, "Deep Volumetric Ambient Occlusion," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1268–1278, 2020.

[17] 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*, vol. 26, no. 1, pp. 23–33, 2019.

[18] 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*, 2019.

[19] Y. Xie, E. Franz, M. Chu, and N. Thuerey, "TempoGAN: A Temporally Coherent, Volumetric GAN for Super-Resolution Fluid Flow," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, pp. 1–15, 2018.

[20] J. Han and C. Wang, "TSR-TVD: Temporal Super-Resolution for Time-Varying Data Analysis and Visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 205–215, 2019.

[21] ——, "SSR-TVD: Spatial Super-Resolution for Time-Varying Data Analysis and Visualization," *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[22] 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 *IEEE Pacific Visualization Symposium (PacificVis)*. IEEE Computer Society, 2020, pp. 71–80.

[23] Y. An, H.-W. Shen, G. Shan, G. Li, and J. Liu, "STSRNet: Deep Joint Space-Time Super-Resolution for Vector Field Visualization," *IEEE Computer Graphics and Applications*, 2021.

[24] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, "Deep Fluids: A Generative Network for Parameterized Fluid Simulations," in *Computer Graphics Forum*, vol. 38, no. 2. Wiley Online Library, 2019, pp. 59–70.

[25] S. Hazarika, H. Li, K.-C. Wang, H.-W. Shen, and C.-S. Chou, "NNVA: Neural Network Assisted Visual Analysis of Yeast Cell Polarization Simulation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 34–44, 2019.

TABLE 8
Mathematical Symbol Table

Symbol	Meaning
P_{sim}	Input simulation parameters
i, j, l, m	Integer indices
F	Function maps input simulation parameters to simulation outputs
S	Simulation outputs
$F_{p\rightarrow ar}$	Function maps input simulation parameters to outputs with adaptive resolutions
$F_{ar\rightarrow s}$	Function converts data with adaptive resolutions to full resolution
L	Number of graph hierarchy levels
$\{G_0, G_1, \ldots, G_{L-1}\}$	Graph hierarchy
$\{V_0, V_1, \ldots, V_{L-1}\}$	Node set list in the graph hierarchy
$\{E_0, E_1, \ldots, E_{L-1}\}$	Edge set list in the graph hierarchy
$E_l^{\leftrightarrow}$	Horizontal edge set in level l
$E_l^{\updownarrow}$	Vertical edge set in level l
$v_{l,i}$	The i^{th} node in level l
$(v_{l,i}, v_{l,j})$	Edge connecting the i^{th} node and the j^{th} node in level l
$(x_{l,i}, y_{l,i}, z_{l,i})$	Cartesian coordinate of node $v_{l,i}$
$(r_{l,i}, \theta_{l,i}, \phi_{l,i})$	Spherical coordinate of node $v_{l,i}$
$d(v_{l,i}, v_{l,j})$	Distance between the i^{th} node and the j^{th} node in level l
$d^{\leftrightarrow}$	Distance between horizontal neighbors (great-circle distance)
$d^{\updownarrow}$	Distance between vertical neighbors (Euclidean distance)
$\rho_l^{\leftrightarrow}$	Average distance square of all the horizontal edges in level l
$\rho_l^{\updownarrow}$	Average distance square of all the vertical edges in level l
$w_l(v_{l,i}, v_{l,j})$	Weight of edge $(v_{l,i}, v_{l,j})$
$\mathcal{D}$	Direction Set
$\lvert\phi\rvert\uparrow, \lvert\phi\rvert\downarrow, West, East, r\uparrow, r\downarrow$	Directions
$\delta_l^x, \delta_l^y, \delta_l^z$ $\delta_l^{lat}, \delta_l^{lon}, \delta_l^h$	Offset between two nodes in level l
$(\gamma_0^{\lvert\phi\rvert\uparrow}, \gamma_0^{\lvert\phi\rvert\downarrow}, \gamma_0^{west}, \gamma_0^{east}, \gamma_0^{r\uparrow}, \gamma_0^{r\downarrow})^{\mathsf{T}}$	Edge attribute vector at level l
$\lvert\lvert \cdot \rvert\rvert$	Euclidean norm
$\times$	Cross product
$\{\mathbf{\Gamma_0}, \mathbf{\Gamma_1}, \ldots, \mathbf{\Gamma_{L-1}}\}$	Edge attribute vector list
$\{G'_0, G'_1, \ldots, G'_{L-1}\}$	Transformed graph hierarchy
$\{\mathbf{\Gamma'_0}, \mathbf{\Gamma'_1}, \ldots, \mathbf{\Gamma'_{L-1}}\}$	Transformed edge attribute vector list
C	Remaining graph hierarchical tree after the cutting
$\psi_C(v_{l,j})$	Proxy of a cut node
N_1, N_2	Number of simulation runs
R, D	Generator, Discriminator
k	Channel multiplier used to control network's size
c, c_{in}, c_{out}	Feature map's channel numbers
Ω_l	Input feature map on graph G'_l
Υ_l	Output feature map on graph G'_l
$\eta(l,i)$	Edge set containing all the edges pointing to node $v_{l,i}$
$\tau(m,l,i)$	Node $v_{m,i}$'s descendant nodes in V'_l
U	Weighted basis matrix in a convolution layer
b	Batch size
$S_{ar}, \hat{S}_{ar}$	Ground Truth and predicted adaptive resolution data

[26] J. Han, H. Zheng, Y. Xing, D. Z. Chen, and C. Wang, "V2V: A Deep Learning Approach to Variable-to-Variable Selection and Translation for Multivariate Time-Varying Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1290–1300, 2020.

[27] Y. Wang, Z. Zhong, and J. Hua, "DeepOrganNet: On-the-Fly Reconstruction and Visualization of 3D/4D Lung Models from Single-View Projections by Deep Deformation Network," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 960–970, 2019.

[28] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv preprint arXiv:1511.06434*, 2015.

[29] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *Proc. International Conference on International Conference on Machine Learning*, vol. 30, no. 1, 2013, p. 3.

[30] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral Normalization for Generative Adversarial Networks," in *Proc. International Conference on Learning Representations*, 2018.

[31] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed Precision Training," in *Proc. International Conference on Learning Representations*, 2018.

[32] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[33] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. International Conference on Learning Representations*. [Online]. Available: http://arxiv.org/abs/1412.6980

[34] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-Attention Generative Adversarial Networks," in *Proc. International Conference on Machine Learning*. PMLR, 2019, pp. 7354–7363.