NUNO: A General Framework for Learning Parametric PDEs with Non-Uniform Data

Songming Liu, Zhongkai Hao, Chengyang Ying,
Hang Su, Ze Cheng, Jun Zhu

Background

Neural Operator

- **Parameterized PDEs**

$$F_a(u(x), x) = 0$$

$a(y) \in \mathcal{A}, y \in \Omega_a$ is the parameter function (e.g., bias term of the boundary condition)

$u(x) \in \mathcal{U}, x \in \Omega_u$ is the unknown solution

- **Training neural operators**
  - To approximate the operator $G^\dagger: \mathcal{A} \to \mathcal{U}$ with a neural model f_θ
  - Data-driven training with a dataset $\{(a_j, u_j)\}_{j=1}^N$, where $a \sim \mu$

$$\min_{\theta \in \Theta} \mathbb{E}_{a\sim\mu} \|f_\theta(a) - u\|$$

Neural Operator

- **Parameterized PDEs**

$$F_a(u(x), x) = 0$$

$a(y) \in \mathcal{A}, y \in \Omega_a$ is the parameter function (e.g., bias term of the boundary condition)

$u(x) \in \mathcal{U}, x \in \Omega_u$ is the unknown solution

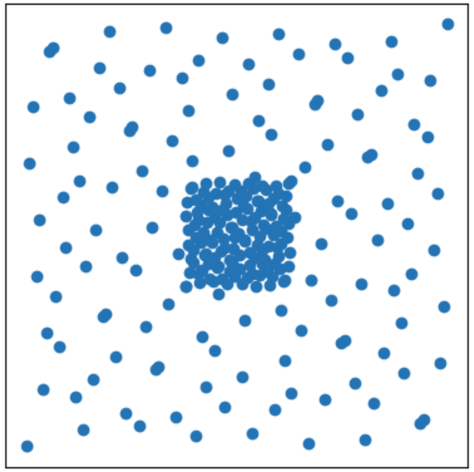
- **Training neural operators**
  - To approximate the operator $G^{\dagger}: \mathcal{A} \to \mathcal{U}$ with a neural model f_θ
  - Data-driven training with a dataset $\{(a_j, u_j)\}_{j=1}^{N}$, where $a \sim \mu$

$$\min_{\theta \in \Theta} \mathbb{E}_{a \sim \mu} \| f_\theta(a) - u \|$$

Challenge of Non-Uniform Data

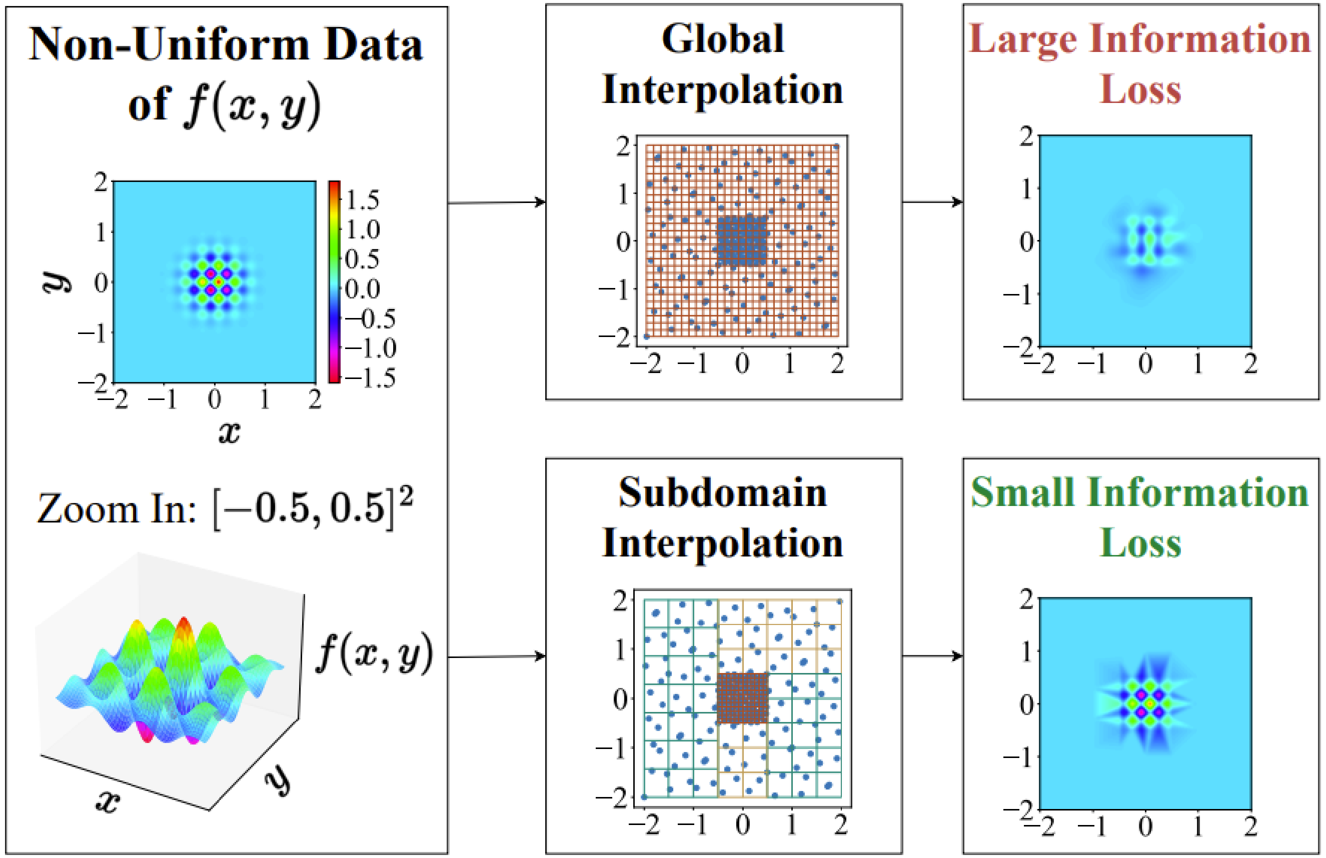
- **Our model f_θ**
  - takes a function $a(y)$ as input and
  - outputs another function $u(x)$ as output
- **Representation of functions**
  - Point-cloud values, $a \coloneqq \{a(x^{(i)})\}_{i=1}^{M}$
- **Challenge**
  - Hard to extract features: efficiency, …
  - Unable to employ **mesh-based** techniques, e.g., FFT, CNN, …

Point cloud $\{x^{(i)}\}_{i=1}^{M}$



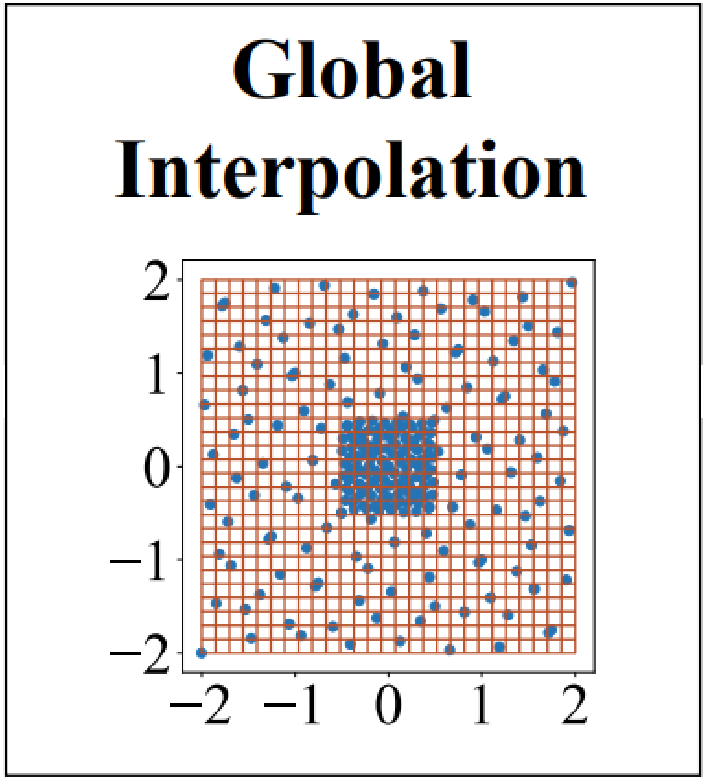
Interpolation Error

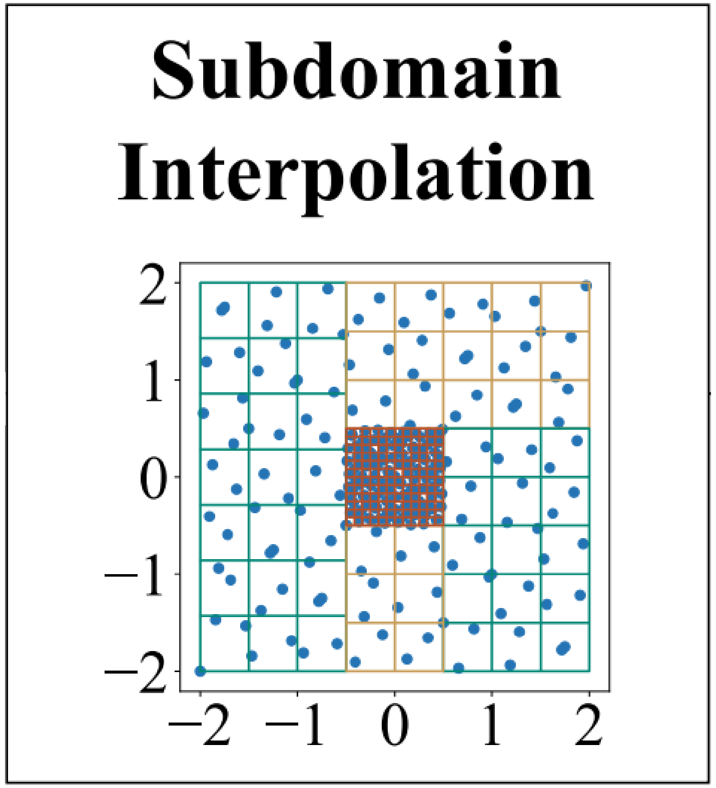
- **Interpolation from point cloud to uniform grids**



Method

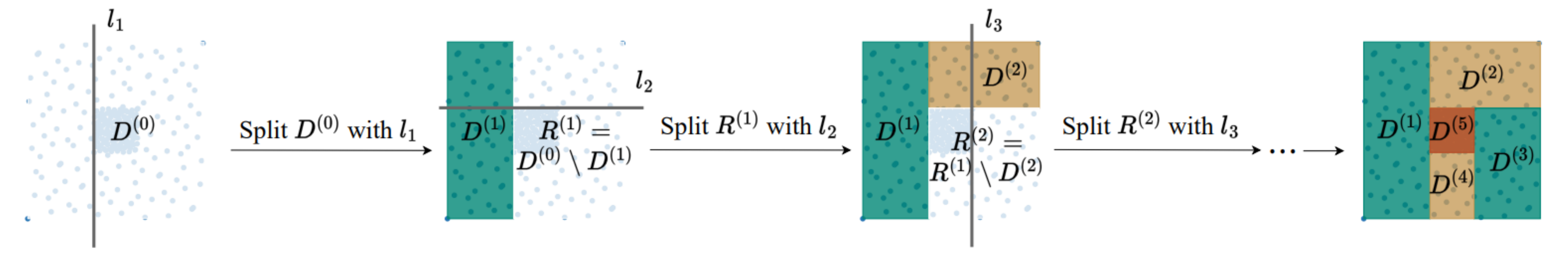
Subdomain Interpolation





- Divide the domain into several **subdomains**
- Interpolation over each subdomain **separately**
- Adaptively apply **coarse** uniform grids on **sparse** subdomains and **fine** grids on **dense** ones

K-D Tree Domain Decomposition



Algorithm 1 Domain Decomposition via a K-D Tree

1: **Input:** initial point cloud $D^{(0)}$ and the number of sub-point clouds n
2: **Output:** a sef of sub-point clouds $\mathcal{S}$
3: **Initialize:** $\mathcal{S} \leftarrow \{D^{(0)}\}$
4: **repeat**
5: Choose $D^* = \arg\max_{D \in \mathcal{S}}(|D| \cdot \mathrm{KL}(P \parallel Q; D))$
6: $\mathcal{S} \leftarrow \mathcal{S} - \{D^*\}$
7: Select the dimension $k, 1 \leq k \leq d$, where the bounding box of D^* has the largest scale
8: Determine the hyperplane $x_k = b^*$, where $b^* = \arg\max_{b \in \mathcal{B}} \mathrm{Gain}(D^*, b)$ and $\mathcal{B}$ is a set of discrete candidates for b
9: Partition D^* with $x_k = b^*$
10: $\mathcal{S} \leftarrow \mathcal{S} \cup \{D^*_{x_k > b^*}, D^*_{x_k \leq b^*}\}$
11: **until** $|\mathcal{S}| = n$

$$\mathrm{KL}(P \parallel Q; D) := \sum_j \frac{|D_j|}{|D|} \ln \left(\frac{|D_j|}{|D|} \Big/ \frac{1}{\prod_{i=1}^d N_i} \right). \tag{4}$$

In calculating P, we employ histogram density estimation. Specifically, we divide the bounding box of D uniformly into $N_1 \times \cdots \times N_d$ cells, and D_j is defined as $\{x \mid x \in D \wedge x \in \text{the } j\text{-th cell}\}$, for $1 \leq j \leq \prod_{i=1}^d N_i$.

$$\begin{aligned} &\mathrm{KL}(P \parallel Q; D^*) - \frac{|D^*_{x_k > b}|}{|D^*|} \mathrm{KL}(P \parallel Q; D^*_{x_k > b}) \\ &\quad - \frac{|D^*_{x_k \leq b}|}{|D^*|} \mathrm{KL}(P \parallel Q; D^*_{x_k \leq b}), \end{aligned} \tag{5}$$

where $D^*_{x_k > b}, D^*_{x_k \leq b}$ are defined as $\{x \mid x \in D^* \wedge x_k > b\}$ and $\{x \mid x \in D^* \wedge x_k \leq b\}$, respectively.

Overall Architecture

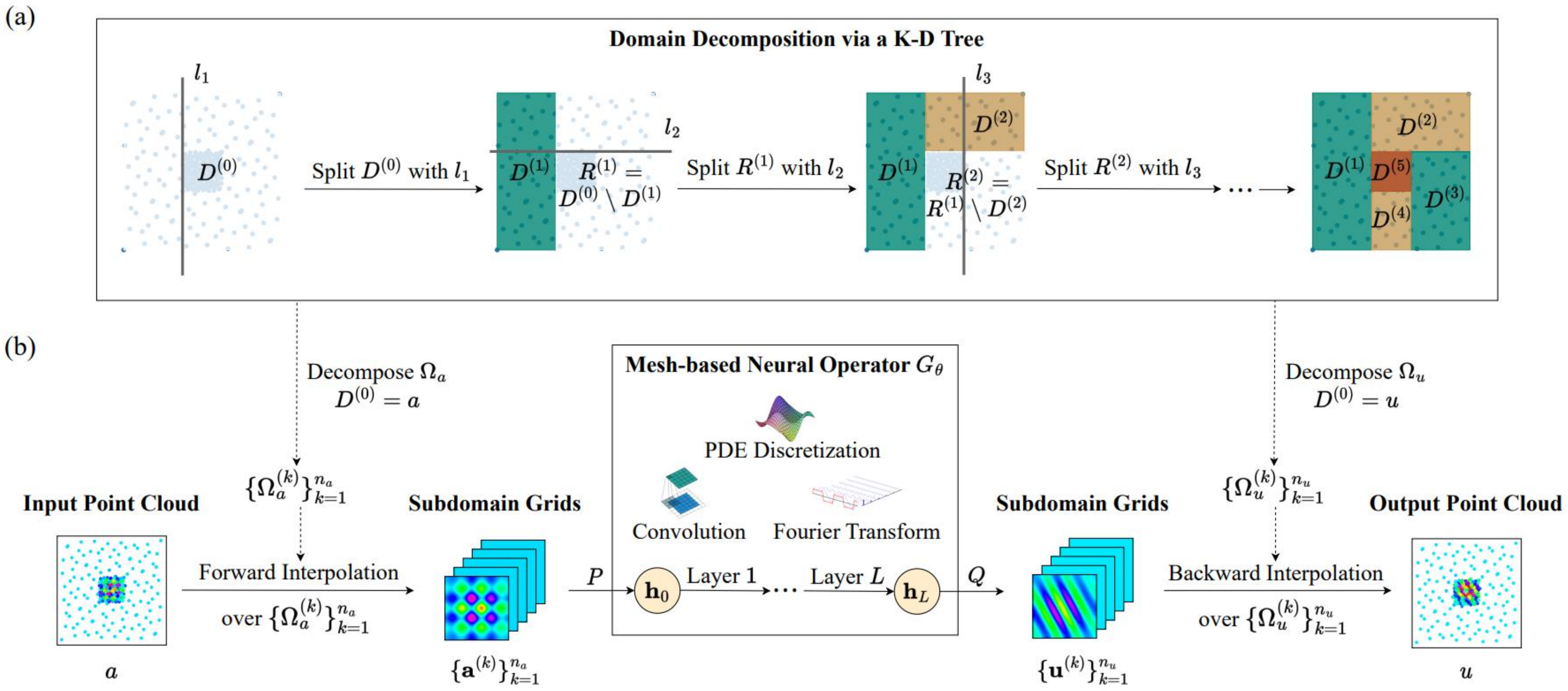


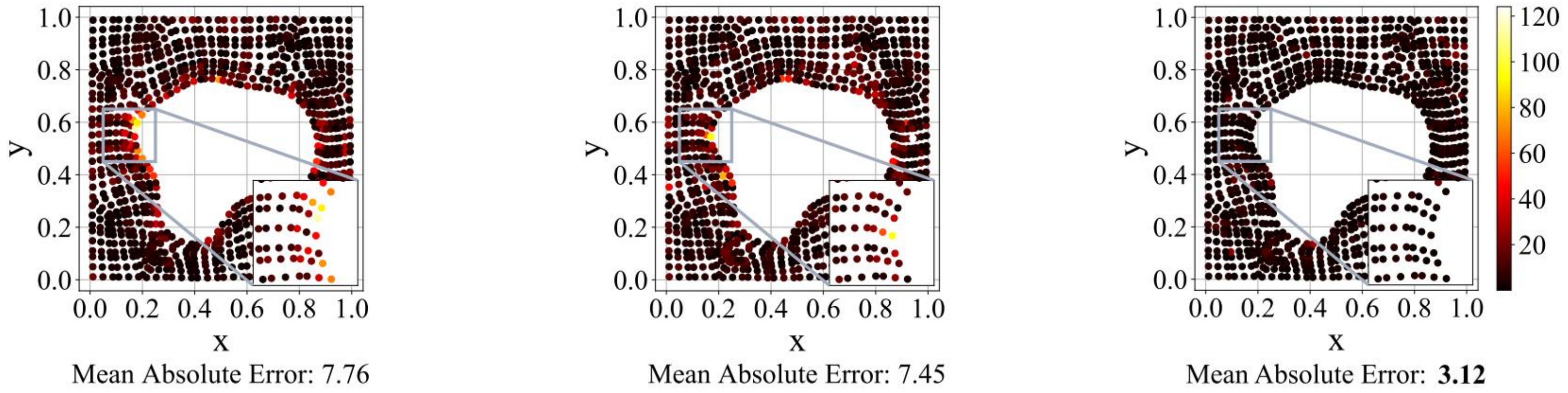
Figure 2: **(a) Domain decomposition:** starting from $D^{(0)}$, each time we choose to split a sub-point cloud with a hyperplane l_i. After 4 iterations, we obtain 5 sub-point clouds distributed more uniformly within their bounding boxes. **(b) NUNO framework:** 1. interpolation to subdomain grids; 2. projection by P; 3. pass through the mesh-based neural operator with L layers ($\mathbf{h}_0, \ldots, \mathbf{h}_L$ indicate hidden embeddings of each layer); 4. projection by Q; 5. interpolation back to the point cloud.

Experiments

Experiment Setup

- **Evaluation Metric:**
  - L^2 Relative Error. We report the mean and 95% CI in 5 runs.
- **Baselines:**
  - **Mesh-based Neural Operators:** U-Net[1], FNO[2], MWNO[3], NU-NO (**ours**, "NU-FNO" indicates that our framework adopts an FNO as the underlying mesh-based model)
  - **Mesh-less Neural Operators:** DeepONet[4], GraphNO[5], Geo-FNO[6], PointNet[7]
- **Problems:**
  - 2D Elasticity, (2+1)D Channel Flow, 3D Heatsink

2D Elasticity



(a) FNO (global interpolation) (b) Geo-FNO (learned) (c) NU-FNO (**ours**)

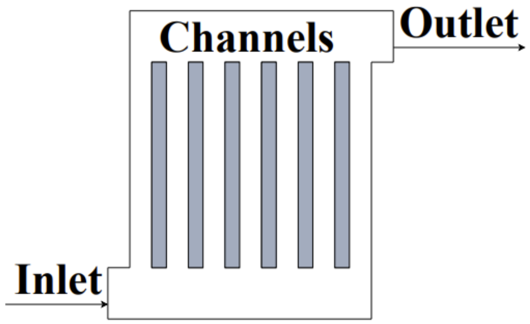
Table 1: Experimental results of 2D elasticity. R mesh and O mesh are adaptive meshes (Li et al., 2022).

Method	Mesh Size	Training Time		L^2 Relative Error ($\times 10^{-2}$)	
		per Epoch	per Run	Training	Testing
NU-FNO (**ours**)	1024[1]	2.3 s	19.6 min	$\mathbf{1.68 \pm 0.08}$	$\mathbf{1.93 \pm 0.11}$
FNO (global interpolation)	1681	**1.2** s	**9.7** min	3.41 ± 0.08	6.16 ± 0.16
U-Net (global interpolation)	1681	2.3 s	18.8 min	1.74 ± 0.02	6.82 ± 0.06
Geo-FNO (R mesh)	1681	1.7 s	14.2 min	3.53 ± 0.09	5.03 ± 0.09
Geo-FNO (O mesh)	1353	2.0 s	16.4 min	4.12 ± 0.16	4.28 ± 0.15
Geo-FNO (learned)	meshless	2.3 s	19.1 min	2.07 ± 0.99	4.31 ± 2.24
GraphNO	meshless	96.8 s	5.4 h	17.5 ± 1.26	16.9 ± 1.28
DeepONet	meshless	40.0 s	11.0 h	2.80 ± 0.13	12.0 ± 0.16

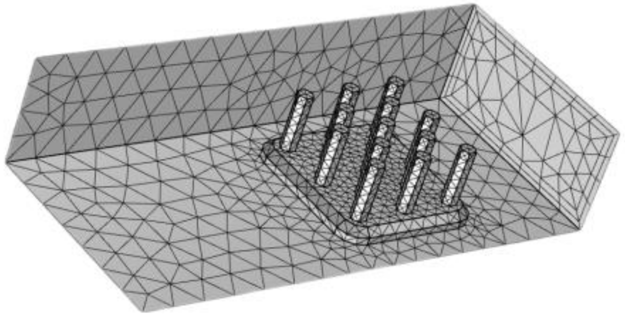
[1] In this paper, all mesh sizes mentioned for our method specifically refer to the *combined total number* of points across all subdomains, rather than the number of points within each individual subdomain.

(2+1)D Channel Flow & 3D Heatsink

- (2+1)D channel flow



- 3D heatsink

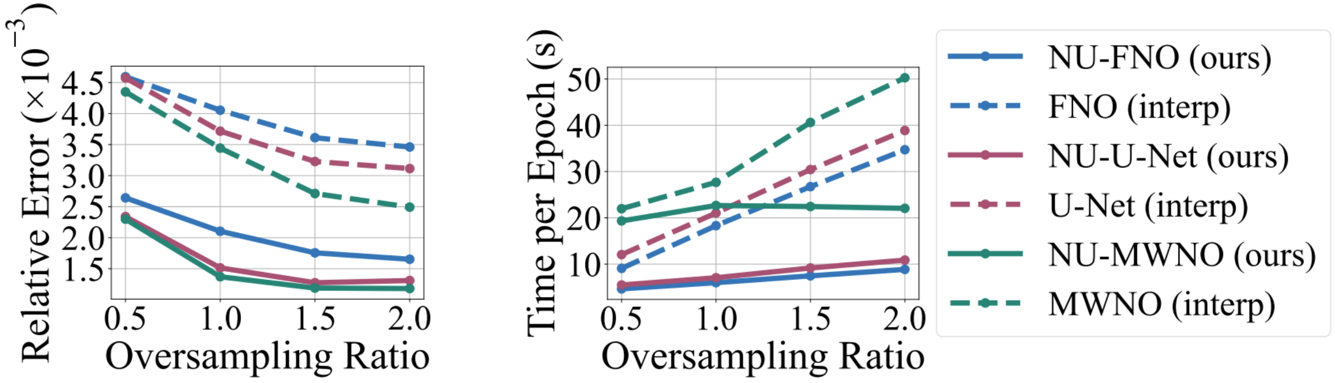


- L^2 relative error and training time per epoch

Methods	(2+1)D Channel flow		3D Heatsink	
	Error ($\times 10^{-3}$) ↓	**Time** ↓	**Error** ($\times 10^{-2}$) ↓	**Time** ↓
NU-FNO (**ours**)	1.74 $\pm$ 0.05	**7.2** s	**5.09** $\pm$ **0.48**	**4.7** s
FNO (global interp)	3.61 $\pm$ 0.19	26.9 s	7.68 $\pm$ 0.25	6.2 s
NU-U-Net (**ours**)	1.27 $\pm$ 0.03	8.6 s	6.31 $\pm$ 0.31	5.2 s
U-Net (global interp)	3.29 $\pm$ 0.15	29.5 s	8.27 $\pm$ 0.14	7.4 s
NU-MWNO (**ours**)	**1.22** $\pm$ **0.06**	22.9 s	5.36 $\pm$ 0.24	18.3 s
MWNO (global interp)	2.71 $\pm$ 0.63	40.6 s	7.38 $\pm$ 0.12	21.0 s
Geo-FNO (learned)	2.15 $\pm$ 0.83	114.9 s	-	-
GraphNO	37.02 $\pm$ 0.00	193.6 s	-	-
DeepONet	119.86 $\pm$ 15.82	235.8 s	-	-
PointNet	-	-	56.42 $\pm$ 27.60	45.3 s

Ablation Study

- How does the performance vary with **mesh size** and the **number of subdomains**?



	Method	Number of Subdomains			
		4	**8**	**12**	**16**
L^2 **Rel. Err.**	NU-FNO	1.92	1.71	1.31	**0.85**
($\times 10^{-3}$)	NU-U-Net	1.33	1.27	1.21	**0.88**
Average	NU-MWNO	1.23	1.19	1.06	**0.72**
Training	NU-FNO	9.5	7.2	6.7	**5.7**
Time per	NU-U-Net	11.2	8.6	8.0	**6.9**
Epoch (s)	NU-MWNO	27.6	23.9	23.1	**22.7**

"Blessing of Subdomains"

References

[1] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention, pp. 234–241. Springer, 2015.

[2] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895, 2020a.

[3] Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. Advances in Neural Information Processing Systems, 34:24048–24062, 2021.

[4] Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. Nature Machine Intelligence, 3(3):218–229, 2021.

References

[5] Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. arXiv preprint arXiv:2003.03485, 2020b.

[6] Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for pdes on general geometries. arXiv preprint arXiv:2207.05209, 2022.

[7] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 652–660, 2017.

Thank You!

Paper

Code