Boundary-Augmented Neural Operators for Better Generalization to Unseen Geometries

Jiayi Zhou
Peking University
jiayi22@pku.edu.cn

Valentin Duruisseaux
California Institute of Technology
vduruiss@caltech.edu

Daniel Zhengyu Huang
Peking University
huangdz@bicmr.pku.edu.cn

Anima Anandkumar
California Institute of Technology
anima@caltech.edu

Abstract

Neural operators have emerged as powerful machine learning models for solving parametric PDEs. For downstream tasks, it is often essential that trained models remain reliable on out-of-distribution samples. However, models trained exclusively on reference data, without additional guidance, can underperform when applied on inputs outside the training distribution. In addition, while boundary conditions can significantly impact PDE solutions, they are often overlooked in existing neural operator designs. In this work, we focus specifically on the challenge of geometry generalization in neural operators and introduce the Boundary-Augmented Neural Operator (BNO), a general framework that incorporates the interaction between the boundary and the full domain. We validate the proposed BNOs on an airfoil flap dataset and a new Poisson equation dataset, in comparison with existing neural operator architectures. In particular, we evaluate a special case of BNO that treats the boundary and full domain separately, thereby retaining efficiency by exploiting the lower dimensionality of boundaries. Our results show that BNOs achieve greater robustness to changes in discretization and point distributions while maintaining high computational efficiency. Furthermore, they handle diverse geometric and topological variations with improved generalization to unseen geometries.

1 Introduction

Solving parametric partial differential equations (PDEs) often requires computationally expensive simulations, particularly for high-dimensional problems. To alleviate this computational cost, neural operators [2, 4] have emerged as promising machine learning models for accelerating PDE-based simulations. Recent advances in neural operator architectures have further extended their applicability to problems with complex geometries. Deformation-based methods, such as Geo-FNO [33], GINO [36, 38], OTNO [32], and others [1, 53], transform input functions defined on the physical space into a structured latent space where more efficient models can be used. Alternatively, methods such as GNPs [46, 47], PCNO [54], and others [13, 39], can operate directly on arbitrary meshes or point clouds in the physical space. A more comprehensive discussion of relevant works is presented in Appendix A.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI4Science.

Despite these advances, existing frameworks are limited in two key aspects:

1. The input and output functions are always defined on the same computational domain, which overlooks an important characteristic of many PDEs: in many practical cases, the solution is significantly influenced by, or inherently dependent on the boundary, rather than the entire domain.
2. Generalization across varying geometries remains challenging. Traditional solvers such as panel methods [5, 14, 44] or boundary element methods (BEM) [10, 49] naturally accommodate arbitrary shapes and topological variations, whereas data-driven neural operator approaches often struggle with out-of-distribution generalization when confronted with unseen geometries. This limitation poses a significant barrier for downstream tasks such as inverse design, where the ability to robustly handle diverse boundary conditions and geometric variations is essential.

Proposed Approach. We introduce **Boundary-augmented Neural Operators (BNOs)**, a general neural operator framework which facilitates the interaction between the boundary and the full computational domain. Motivated by the theoretical foundations of boundary integral equations, we consider a special case in which the boundary and full-domain features are treated separately and coupled via extension and reduction operators. This allows direct prediction of full-field solution operators from boundary and leverages the lower dimensionality of boundary submanifolds to improve computational efficiency and reduce memory requirements in large-scale simulations.

We focus especially on evaluating the geometric generalization of neural operators on a fixed PDE system across previously unseen geometries and topologies, rather than solely on test errors on fixed classes of geometries. To facilitate systematic benchmarking under varying geometric and topological conditions, we also construct a new Poisson equation dataset of diverse shapes.

Summary of results. The proposed BNO demonstrates strong robustness to changes in the discretization and distribution of points. BNO can effectively handle topological and geometric variations. In particular, we train BNO models on a relatively small airfoil dataset (with two different topologies, Airfoil and Airfoil Flap) and Poisson dataset (with four different types of geometries and topologies), and achieve relative L_2 test errors around 2.0% and 1.7%, respectively. In addition, we evaluate the model's potential ability to generalize across previously unseen geometries. On the airfoil dataset, when trained solely on Airfoil and tested on Airfoil Flap, the model still produces physically reasonable solutions. In the Laplace dataset, where the training and testing sets correspond to different geometries and topologies, the model also generalize well. Moreover, the training process is highly efficient thanks to the reduction in dimensionality, with the time per epoch being less than half that of the PCNO baselines.

2 Problem Setting

2.1 Boundary Value Problem

Let $D \subset \mathbb{R}^n$, and let $\Omega_\alpha \subset D$ be compact regions parameterized by some $\alpha \in \mathcal{A}$. Given partial differential operators $\mathcal{P}$ and $\mathcal{R}_\alpha$, a class of partial differential equations is given by the boundary value problem

$$\begin{cases} \mathcal{P}u = f & \text{in } \Omega_\alpha, \\ \mathcal{R}_\alpha u = g & \text{on } \partial\Omega_\alpha. \end{cases}$$

Here, the *tangential* differential operator $\mathcal{R}_\alpha$ only acts on u and its partial derivatives with respect to the normal ν_α on the boundary $\partial\Omega_\alpha$ of the domain Ω_α. A natural interpretation of the input functions f and g is that they belong to spaces defined on the domain Ω_α and its boundary $\partial\Omega_\alpha$, respectively.

We use the notation $\mathscr{F}(Y)$ to denote a class of functions defined on a general set Y. For instance, one may assume $f \in L^2(\Omega_\alpha)$ and $g \in L^2(\partial\Omega_\alpha)$. By extending f, g, u using zeros to the entire domain D, the spaces $\mathscr{F}(\Omega_\alpha)$ and $\mathscr{F}(\partial\Omega_\alpha)$ can be viewed as embedded subspaces of $\mathscr{F}(D)$, which we denote by $\mathcal{F}_{\text{int}}$ and $\mathcal{F}_{\text{b}}$, respectively. Similarly, the solution space is represented by $\mathcal{U}$. Moreover, the geometric parameter α can also be represented as a function over D, such as a signed distance function (SDF) [36] or a density field [32, 54].

Together, this leads to a unified formulation of the solution operator

$$\mathcal{G}^\dagger : \mathcal{A} \times \mathcal{F}_{\text{int}} \times \mathcal{F}_{\text{b}} \to \mathcal{U}, \qquad \mathcal{G}^\dagger : (\alpha, f, g) \mapsto u. \tag{1}$$

A more general formulation can be obtained without extending the functions, where the solution operator is formulated as

$$\mathcal{G}^\dagger : \bigcup_{\alpha \in \mathcal{A}} (\{\alpha\} \times \mathscr{F}(\Omega_\alpha) \times \mathscr{F}(\partial\Omega_\alpha)) \to \bigcup_{\alpha \in \mathcal{A}} \mathscr{F}(\Omega_\alpha), \qquad \mathcal{G}^\dagger : (\alpha, f, g) \mapsto u. \tag{2}$$

In this paper, we retain the form of equation (2) to emphasize that the functions f and g are defined on manifolds of different dimensions.

2.2 Boundary Integral Equations

Using the General Green Theorem (see [23, equation (3.5.19)]), we can derive the weak formulation of the original equation,

$$\int_{\Omega_\alpha} u\mathcal{P}^*\phi \, dy = \int_{\Omega_\alpha} \phi f \, dy + \int_{\partial\Omega_\alpha} \mathcal{B}(u, \phi, \alpha) \, dS(y), \tag{3}$$

where $\mathcal{P}^*$ is the adjoint operator of $\mathcal{P}$ and $\mathcal{B}(u, \phi, \alpha)$ denotes the boundary terms, depending on the solution u, the parameters α, and a given test function ϕ.

For example, for the Laplace equation,

$$\mathcal{P}u = \Delta u \quad \text{and} \quad \mathcal{B}(u, \phi, \alpha) = u\frac{\partial \phi}{\partial \nu_\alpha} - \phi\frac{\partial u}{\partial \nu_\alpha},$$

where ν_α denotes the normal on the boundary $\partial\Omega_\alpha$, as before. Furthermore, if $\mathcal{P}$ is a uniformly strongly elliptic differential operator of even order with real coefficients, Lemma 3.6.1 in [23] shows that there exists a local fundamental solution $G(x, y)$, such that $\mathcal{P}^*G(x, y) = \delta(y - x)$, and thus the solution can be expressed as

$$u(x) = \int_{\Omega_\alpha} G(x, y) f(y) \, dy + \int_{\partial\Omega_\alpha} \mathcal{B}(u, G, \alpha)(y) \, dS(y) \qquad \text{for } x \in \text{int } \Omega_\alpha. \tag{4}$$

3 Boundary-Augmented Neural Operators

3.1 General Approach

We formulate the **Boundary-augmented Neural Operator** (BNO) as a three-stage architecture consisting of a lifting operator, a sequence of boundary-augmented layers, and a projection operator,

$$(Lifting) \qquad (u_0, v_0) = P(f, g) \tag{5}$$

$$(Boundary\text{-}Augmented\ Layers) \qquad (u_\ell, v_\ell) = \mathcal{L}(u_\ell, v_\ell) \quad \text{for } \ell = 1, \ldots, L \tag{6}$$

$$(Projection) \qquad (u, v) = Q(u_L, v_L) \tag{7}$$

where f, g are the input features that defines on Ω_α and $\partial\Omega_\alpha$, separately. Similarly, u_ℓ is defined over the full domain Ω_α, while v_ℓ is defined only on the boundary $\partial\Omega_\alpha$.

The boundary-augmented layers require interactions between u_ℓ and v_ℓ, i.e., between the full domain and boundary. This can be implemented using a boundary encoder that couples the two (see Appendix A). However, in the special case considered here, the boundary and full-domain features are treated separately, which allows to exploit the lower dimensionality of the boundary to improve computational efficiency while still capturing the essential boundary effects.

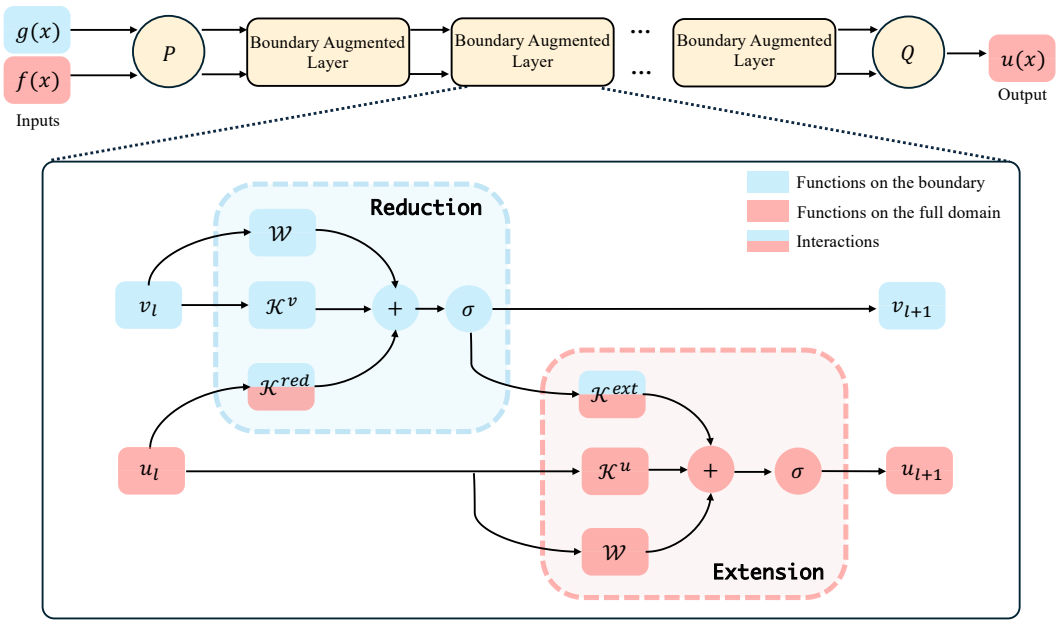

Figure 1: Example of a Boundary-augmented Neural Operator (BNO). The blue part indicates operators' action on functions defined on the boundary, while the red part corresponds to the operators acting on functions over the full domain. The mixed colors of blue and red illustrate that for this specific BNO architecture, $\mathcal{K}^{\text{red}}$ and $\mathcal{K}^{\text{ext}}$ are interactions between the boundary and the full domain.

3.2 Special Case: Reduction and Extension

In this special case, we exploit the boundary integral formulation introduced above and focus specifically on the boundary term $\mathcal{B}(u, \phi, \alpha)$ which depends on the solution u, the parameters α, and any given test function ϕ.

As an example, for the 2D Poisson equation $\Delta u = f$, we can use the notation $\nabla\phi = (\phi_x, \phi_y)$, $\nabla u = (u_x, u_y)$, $\partial\nu_\alpha = (\nu_\alpha^x, \nu_\alpha^y)$ and rewrite the boundary term

$$\mathcal{B}(u, \phi, \alpha) = u\frac{\partial\phi}{\partial\nu_\alpha} - \phi\frac{\partial u}{\partial\nu_\alpha}.$$

as

$$\begin{aligned}\mathcal{B}(u, \phi, \alpha) &= u\phi_x\nu_\alpha^x + u\phi_y\nu_\alpha^y + u_x\phi\nu_\alpha^x + u_y\phi\nu_\alpha^y \\ &= (\phi_x, \phi_y, \phi, \phi)\cdot(u\nu_\alpha^x,\, u\nu_\alpha^y,\, u_x\nu_\alpha^x,\, u_y\nu_\alpha^y)^\top \\ &=: \phi_{\mathcal{B}}\cdot v\end{aligned} \tag{8}$$

where the term $\phi_{\mathcal{B}}$ is independent of the geometry, while v depends on the geometry.

More generally, it can be shown (see Appendix B) that the boundary term $\mathcal{B}(u, \phi, \alpha)$ can be decomposed into such a geometry-dependent part and a geometry-independent part. Using the same notations for $\phi_{\mathcal{B}}$ and v, equation (4) can be rewritten as

$$u(x) = \int_{\Omega_\alpha} G(x,y)f(y)\,dy + \int_{\partial\Omega_\alpha} G_{\mathcal{B}}(x,y)\cdot v(u,\alpha)(y)\,dS(y) \qquad \text{for } x \in \text{int }\Omega_\alpha. \tag{9}$$

Inspired by the structure of equation (9), we propose the following special case of BNO, which is also illustrated in Figure 1:

$$(Lifting) \qquad (u_0, v_0) = P(f, g) \tag{10}$$

$$(Reduction) \qquad v_\ell = \sigma\big(\mathcal{K}_\ell^v v_{\ell-1} + \mathcal{K}_\ell^{\text{red}} u_{\ell-1} + \mathcal{W}_\ell^v v_{\ell-1}\big) \quad \text{for } \ell = 1, \dots, L \tag{11}$$

$$(Extension) \qquad u_\ell = \sigma\big(\mathcal{K}_\ell^u u_{\ell-1} + \mathcal{K}_\ell^{\text{ext}} v_\ell + \mathcal{W}_\ell^u u_{\ell-1}\big) \quad \text{for } \ell = 1, \dots, L \tag{12}$$

$$(Projection) \qquad u = Qu_L \tag{13}$$

where f, u_ℓ are defined over the full domain Ω_α, and g, v_ℓ is defined only on the boundary $\partial\Omega_\alpha$.

In this special case of Boundary-augmented Neural Operators,

- P and Q are pointwise **lifting and projection operators** to extract and project high-dimensional features, and $\mathcal{W}$ is a pointwise linear operator, as in the traditional structure of neural operators.

- $\mathcal{K}_\ell^u$ and $\mathcal{K}_\ell^v$ correspond to conventional **neural operator layers** (e.g., Fourier convolution layers) applied either on the full domain Ω_α or on the boundary $\partial\Omega_\alpha$ with the dimensions of input and output space remaining the same.

- $\mathcal{K}_\ell^{\text{red}} : \mathcal{L}(\Omega_\alpha; \mathbb{R}^d) \to \mathcal{L}(\partial\Omega_\alpha; \mathbb{R}^d)$ is a **reduction operator** to obtain v as in equation (9), so it must be a local operator depending solely on the values of $u_{\ell-1}(x)$ in the vicinity of the boundary. It can be represented by a simple differential operator, such as ∇ or $\frac{\partial}{\partial n}$, since the presence of multiple layers implies the use of higher-order derivatives. This operator can be computed directly by least square estimation [54] or parameterized through a local neural operator [40].

- $\mathcal{K}_\ell^{\text{ext}} : \mathcal{L}(\partial\Omega_\alpha; \mathbb{R}^d) \to \mathcal{L}(\Omega_\alpha; \mathbb{R}^d)$ is an **extension operator** designed to emulate the integral formulation in equation (9), which is an integral operator that extends a function defined on $\partial\Omega_\alpha$ to a function defined on Ω_α, which plays a central role in our method.

Here, we follow the common practice in operator learning, and assume that the boundary kernel admits a low-rank representation

$$G_{\mathcal{B}}(x, y) = \sum_k c_k \varphi_k(x) \overline{\varphi_k(y)}, \tag{14}$$

in which case the action of the extension integral operator can be approximated by

$$\mathcal{K}_\ell^{\text{ext}} v \approx \sum_k c_k \varphi_k(x) \left(\int_{\partial\Omega_\alpha} v(y) \overline{\varphi_k(y)}\, dS(y) \right) \tag{15}$$

In practice, the coefficients of v on the boundary $\partial\Omega_\alpha$ can be numerically estimated via

$$\mathcal{F}_{\partial\Omega_\alpha} v(k) := \int_{\partial\Omega_\alpha} v(y) \overline{\varphi_k(y)}\, dS(y) \approx \sum_j v(y_j) \overline{\varphi_k(y)} \Delta S(y_j) \tag{16}$$

when surface elements $\Delta S(y_j)$ are known. In our numerical experiments, we employ a truncated Fourier basis, namely $\left\{ \varphi_k(x) = e^{-i \frac{2\pi k}{L} \cdot x} : k \in [-K, K]^d \right\}$ where $L > 0$ is a learnable scaling parameter that adapts the frequency resolution to the underlying geometry.

Depending on the problem setting, and in particular on whether the solution depends primarily on the full field or only on the boundary, different choices can be made for the reduction operator $\mathcal{K}^{\text{red}}$, the extension operator $\mathcal{K}^{\text{ext}}$, or the global operator components $\mathcal{K}^u, \mathcal{K}^v$ within this example of BNO. The main computational cost typically arises from $\mathcal{K}^u$ and from the reduction operator. When the solution is determined solely by the boundary, as in our numerical experiments, the global operator $\mathcal{K}^u$ on Ω_α can be omitted, which reduces both computational time and memory requirements.

In this paper, we consider two different examples of Boundary-Augmented Neural Operators (BNOs):

- **GNO-BNO** where a Graph Neural Operator (GNO) [34] is used for the reduction step to capture local effects,

- **ExtBNO** where the reduction operator is omitted entirely to achieve further savings in computation, leading to a purely extension-based model.

Table 1: Relative L_2 error (%) and training time per epoch for different models. The top row specifies the training and testing types of airfoil. "Mixed" denotes a combination of the two types of airfoil.

Training Data	Mixed (500×2)	*Airfoil* (1000)		*Airfoil+Flap* (1000)		Time
Testing Data	Mixed	*Airfoil*	*Airfoil+Flap*	*Airfoil*	*Airfoil+Flap*	
ExtBNO	2.07	1.58	11.21	10.86	1.63	17s
GNOBNO	2.03	1.56	**10.92**	**9.97**	**1.58**	24s
PCNO	1.87	1.58	21.74	10.94	1.64	96s
PCFNO	**1.86**	**1.52**	26.10	12.47	1.63	37s

4 Numerical Experiments

In this section, we evaluate the performance of Boundary-Augmented Neural Operators, and more specifically of GNO-BNO and ExtNO, on two PDE problems: (1) the transonic Euler equations on airfoils with and without flaps, and (2) the Laplace equation with Dirichlet boundary conditions over four different types of geometries.

As baselines, we consider the Point Cloud Neural Operator (PCNO) [54], which learns operators directly on unstructured point clouds and incorporates gradient features. We also include PCFNO, a simplified variant of PCNO without gradient features, which can also be regarded as a point-cloud implementation of the Fourier Neural Operator (FNO) [17, 35].

4.1 Airfoil With Flap

The Airfoil-Flap dataset of [54] consists of simulations of the following 2D Euler equations

$$\begin{aligned} \nabla \cdot (\rho \boldsymbol{v}) &= 0, \\ \nabla \cdot (\rho \boldsymbol{v} \otimes \boldsymbol{v} + pI) &= 0, \\ \nabla \cdot ((E+p)\boldsymbol{v}) &= 0, \end{aligned} \tag{17}$$

over two types of shapes with different topologies, a single airfoil (*Airfoil*), and a main airfoil with an additional flap (*Airfoil+Flap*), as depicted in Figure 2. The steady-state solution is determined by the airfoil surface. Therefore, predicting the pressure field can be viewed as mapping from a 1D submanifold (the surface) to the full field, which is well-suited for BNOs.

Figure 2: Left: *Airfoil* example. Right: *Airfoil+Flap* example.

We first train ExtBNO and GNOBNO on a mixed dataset comprising 500 samples from *Airfoil* and 500 samples from *Airfoil+Flap*. The resulting relative L2 test errors of the models are 2.07% and 2.03%, respectively, slightly higher than those reported for PCNO. We then train the models separately on 1,000 samples of either *Airfoil* or *Airfoil+Flap*, and results are reported in Table 1. We find that when trained only on *Airfoil* and tested on *Airfoil+Flap*, BNOs demonstrate significantly better generalization performance, capturing the shock over the main airfoil, whereas PCNO fails to produce physically consistent predictions, as shown in Figure 4.

Table 2: Relative L_2 error (%) and training time per epoch. The left column indicates the training dataset, where "Mixed" denotes a combination of all four types of shape. The top row indicates the test type of shape.

Train\Test	*Low-Freq*	*High-Freq*	*Double*	*Hole*	Time per Epoch
Mixed (512×4)	1.45	2.07	2.02	1.42	61s
Mixed (128×4)	3.30	3.58	4.30	2.91	18s
Low-Freq (512)	2.80	10.44	5.42	7.51	18s
High-Freq (512)	7.88	2.95	7.03	11.64	18s
Double (512)	4.92	8.72	4.09	5.39	18s
Hole (512)	9.96	10.36	8.21	2.48	18s

4.2 Laplace Equation

The Laplace equation with Dirichlet boundary conditions can be expressed as

$$\begin{aligned} \Delta u &= 0, \\ u|_{\partial\Omega} &= g, \end{aligned} \tag{18}$$

where the solution u is determined by the shape of the domain and the boundary values. Four types of geometries in 2D Euclidean space as shown in Figure 3 are considered:

1. *Low-Freq*, a simple connected boundary with low-frequency features,
2. *High-Freq*, a simple connected boundary with high-frequency features,
3. *Double*, two low-frequency boundaries connected together,
4. *Hole*, a shape with a hole.

For each type of geometry, we generate 1024 different random shapes, along with corresponding random functions g. More details regarding the dataset are provided in Appendix D.1.

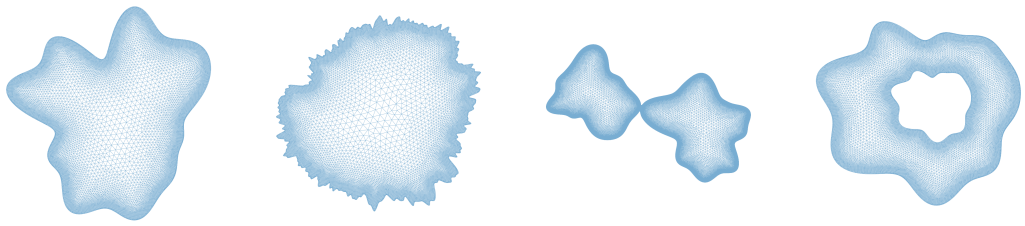

Figure 3: The shapes considered for the Laplace equation. The shapes from left to right correspond to *Low-Freq*, *High-Freq*, *Double*, and *Hole* geometries.

We first train our model on a mixture of all four shape types. For each type, we use 512 samples for training and 100 samples for testing. The resulting test relative L_2 error is 1.74%, indicating that the model's expressive power is sufficient to handle diverse geometries. We then train the model on each type of shape respectively to investigate the generalization properties of the model. The testing errors are shown in Table 2. For the cases where the training and testing datasets are completely disjoint, the model also generalizes well. It is also noteworthy that the model still exhibits potential generalization between the *Hole* shapes and the other shapes, even though the *Hole* geometry is not topologically homeomorphic to the simply connected ones. Some samples from different training settings are shown in D.2.

5 Discussion

The experiments show that by separating the processes of reduction and extension and leveraging the lower dimensionality of the boundary, the considered BNOs achieve much higher efficiency when the solution strongly depends on the boundary. These BNOs also capture the interaction between the boundary and the full domain, leading to improved generalization across unseen topologies and geometries. Several interesting future avenues for research include:

1. Our experiments have been limited to 2D settings. Extending BNOs to 3D large-scale simulations, including time-dependent problems, could be an important next step.
2. Constructing a broader training dataset with a wider variety of shapes would allow the model to encounter more diverse geometries during training and could also serve as a standardized benchmark for future studies.
3. Investigate the structure of interactions between boundary and full domain. Our current design separates reduction and extension, and a direction is to devise coupling mechanisms that preserve the boundary's lower dimensionality while enabling richer mixing with domain features.

References

[1] Zan Ahmad, Shiyi Chen, Minglang Yin, Avisha Kumar, Nicolas Charon, Natalia Trayanova, and Mauro Maggioni. 2024. Diffeomorphic Latent Neural Operators for Data-Efficient Learning of Solutions to Partial Differential Equations. https://doi.org/10.48550/arXiv.2411.18014 arXiv:2411.18014 [cs]

[2] Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. 2024. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics* (2024), 1–9.

[3] Filipe De Avila Belbute-Peres, Thomas Economon, and Zico Kolter. 2020. Combining Differentiable PDE Solvers and Graph Neural Networks for Fluid Flow Prediction. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2402–2411.

[4] Julius Berner, Miguel Liu-Schiaffini, Jean Kossaifi, Valentin Duruisseaux, Boris Bonev, Kamyar Azizzadenesheli, and Anima Anandkumar. 2025. Principled Approaches for Extending Neural Architectures to Function Spaces for Operator Learning. arXiv:2506.10973 [cs.LG] https://arxiv.org/abs/2506.10973

[5] Lothar Birk. 2021. A comprehensive and practical guide to the Hess and Smith constant source and dipole panel. *Ship Technology Research* 69 (09 2021), 1–13. https://doi.org/10.1080/09377255.2021.1966575

[6] Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. 2023. Spherical Fourier Neural Operators: Learning Stable Dynamics on the Sphere. https://doi.org/10.48550/arXiv.2306.03838 arXiv:2306.03838 [cs]

[7] Florent Bonnet, Jocelyn Ahmed Mazari, Paola Cinnella, and Patrick Gallinari. 2022. AIRFRANS: High Fidelity Computational Fluid Dynamics Dataset for Approximating Reynolds-Averaged Navier-stokes Solutions. In *Proceedings of the 36th International Conference on Neural Information Processing Systems (NIPS '22)*. Curran Associates Inc., Red Hook, NY, USA, 23463–23478.

[8] Nicolas Boullé, Christopher J. Earls, and Alex Townsend. 2022. Data-Driven Discovery of Green's Functions with Human-Understandable Deep Learning. *Scientific Reports* 12, 1 (March 2022), 4824. https://doi.org/10.1038/s41598-022-08745-5

[9] Sumanth Kumar Boya and Deepak Subramani. 2024. A Physics-Informed Transformer Neural Operator for Learning Generalized Solutions of Initial Boundary Value Problems. https://doi.org/10.48550/arXiv.2412.09009 arXiv:2412.09009 [cs]

[10] Carlos Alberto Brebbia, José Claudio Faria Telles, and Luiz C Wrobel. 2012. *Boundary element techniques: theory and applications in engineering*. Springer Science & Business Media.

[11] Mateus Mussi Brugnolli, Leonardo Correia, Bruno A. Angélico, and João F. Justo. 2025. Hyper Boundary Conditions: Data-driven Operator for Boundary Value Problems. *Engineering Applications of Artificial Intelligence* 142 (Feb. 2025), 109913. https://doi.org/10.1016/j.engappai.2024.109913

[12] Scott Alexander Cameron, Arnu Pretorius, and Stephen J. Roberts. 2023. Nonparametric Boundary Geometry in Physics Informed Deep Learning. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

[13] Gengxiang Chen, Xu Liu, Qinglu Meng, Lu Chen, Changqing Liu, and Yingguang Li. 2023. Learning Neural Operators on Riemannian Manifolds. arXiv:2302.08166 [cs, math]

[14] Russell M. Cummings, William H. Mason, Scott A. Morton, and David R. McDaniel. 2015. *Applied Computational Aerodynamics: A Modern Engineering Approach*. Cambridge University Press.

[15] Jingyang Deng, Xingjian Li, Haoyi Xiong, Xiaoguang Hu, and Jinwen Ma. 2024. Geometry-Guided Conditional Adaptation for Surrogate Models of Large-Scale 3D PDEs on Arbitrary Geometries. In *Proceedings of the Thirty-ThirdInternational Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Jeju, South Korea, 5790–5798. https://doi.org/10.24963/ijcai.2024/640

[16] Zhiliang Deng, Qinglu Meng, Yingguang Li, Xu Liu, Gengxiang Chen, Lu Chen, Changqing Liu, and Xiaozhong Hao. 2025. BV-NORM: A Neural Operator Learning Framework for Parametric Boundary Value Problems on Complex Geometric Domains in Engineering. *Engineering Applications of Artificial Intelligence* 144 (March 2025), 110109. https://doi.org/10.1016/j.engappai.2025.110109

[17] Valentin Duruisseaux, Jean Kossaifi, and Anima Anandkumar. 2025. Fourier Neural Operators Explained: A Practical Perspective. arXiv:2512.01421 [cs.LG] https://arxiv.org/abs/2512.01421

[18] Lawrence C Evans. 2022. *Partial differential equations*. Vol. 19. American Mathematical Society.

[19] Zhiwei Fang, Sifan Wang, and Paris Perdikaris. 2023. Learning Only On Boundaries: A Physics-Informed Neural Operator for Solving Parametric Partial Differential Equations in Complex Geometries. https://doi.org/10.48550/arXiv.2308.12939 arXiv:2308.12939 [cs]

[20] Adarsh Ganeshram, Haydn Maust, Valentin Duruisseaux, Zongyi Li, Yixuan Wang, Daniel Leibovici, Oscar Bruno, Thomas Hou, and Anima Anandkumar. 2025. FC-PINO: High Precision Physics-Informed Neural Operators via Fourier Continuation. arXiv:2211.15960 [cs.LG]

[21] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. Meshcnn: a network with an edge. *ACM Transactions on Graphics (ToG)* 38, 4 (2019), 1–12.

[22] Junyan He, Seid Koric, Diab Abueidda, Ali Najafi, and Iwona Jasiuk. 2024. Geom-DeepONet: A Point-Cloud-Based Deep Operator Network for Field Predictions on 3D Parameterized Geometries. *Computer Methods in Applied Mechanics and Engineering* 429 (Sept. 2024), 117130. https://doi.org/10.1016/j.cma.2024.117130

[23] George C. Hsiao and Wolfgang L. Wendland. 2021. *Boundary Integral Equations*. Applied Mathematical Sciences, Vol. 164. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-030-71127-6

[24] Daniel Zhengyu Huang, Nicholas H. Nelsen, and Margaret Trautner. 2025. An Operator Learning Perspective on Parameter-to-Observable Maps. *Foundations of Data Science* 7, 1 (2025), 163–225. https://doi.org/10.3934/fods.2024037 arXiv:2402.06031 [cs]

[25] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D. D. Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. 2021. UV-Net: Learning from Boundary Representations. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 11698–11707. https://doi.org/10.1109/CVPR46437.2021.01153

[26] TN Kipf. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907* (2016).

[27] Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Valentin Duruisseaux, Robert Joseph George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, and Anima Anandkumar. 2025. A Library for Learning Neural Operators. arXiv:2412.10354 [cs.LG] https://arxiv.org/abs/2412.10354

[28] Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. 2021. On Universal Approximation and Error Bounds for Fourier Neural Operators. *J. Mach. Learn. Res.* 22, 1, Article 290 (2021).

[29] Hongyu Li, Ximeng Ye, Peng Jiang, Guoliang Qin, and Tiejun Wang. 2023. Local Neural Operator for Solving Transient Partial Differential Equations on Varied Domains. https://doi.org/10.48550/arXiv.2203.08145 arXiv:2203.08145 [cs]

[30] Ruoyan Li, Wenjing Ye, and Yijun Liu. 2025. A Boundary-Based Fourier Neural Operator (B-FNO) Method for Efficient Parametric Acoustic Wave Analysis. *Engineering with Computers* (Jan. 2025). https://doi.org/10.1007/s00366-024-02103-x

[31] Xinyi Li, Zongyi Li, and Nikola Kovachki. 2025. Geometry Operator Learning with Optimal Transport. (2025).

[32] Xinyi Li, Zongyi Li, Nikola Kovachki, and Anima Anandkumar. 2025. Geometric Operator Learning with Optimal Transport. arXiv:2507.20065 [cs.LG] https://arxiv.org/abs/2507.20065

[33] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. 2023. Fourier Neural Operator with Learned Deformations for PDEs on General Geometries. *J. Mach. Learn. Res.* 24, 1 (Jan. 2023), 388:18593–388:18618.

[34] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485* (2020).

[35] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2021. Fourier Neural Operator for Parametric Partial Differential Equations. arXiv:2010.08895 [cs, math]

[36] Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. 2023. Geometry-Informed Neural Operator for Large-Scale 3D PDEs. In *Thirty-Seventh Conference on Neural Information Processing Systems*.

[37] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. 2024. Physics-Informed Neural Operator for Learning Partial Differential Equations. *ACM / IMS Journal of Data Science* 1, 3 (July 2024), 1–27. https://doi.org/10.1145/3648506

[38] Ryan Y. Lin, Julius Berner, Valentin Duruisseaux, David Pitt, Daniel Leibovici, Jean Kossaifi, Kamyar Azizzadenesheli, and Anima Anandkumar. 2025. Enabling Automatic Differentiation with Mollified Graph Neural Operators. https://doi.org/10.48550/arXiv.2504.08277 arXiv:2504.08277 [cs]

[39] Levi Lingsch, Mike Y. Michelis, Emmanuel de Bezenac, Sirani M. Perera, Robert K. Katzschmann, and Siddhartha Mishra. 2024. Beyond Regular Grids: Fourier-Based Neural Operators on Arbitrary Domains. arXiv:2305.19663 [cs]

[40] Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. 2024. Neural Operators with Localized Integral and Differential Kernels. https://doi.org/10.48550/arXiv.2402.16845 arXiv:2402.16845 [cs]

[41] Winfried Lötzsch, Simon Ohler, and Johannes Otterbach. 2022. Learning the Solution Operator of Boundary Value Problems Using Graph Neural Networks. In *ICML 2022 2nd AI for Science Workshop*.

[42] Bin Meng, Yutong Lu, and Ying Jiang. 2024. Solving Partial Differential Equations in Different Domains by Operator Learning Method Based on Boundary Integral Equations. https://doi.org/10.48550/arXiv.2406.02298 arXiv:2406.02298 [math-ph]

[43] Monika Nagy-Huber and Volker Roth. 2024. Physics-Informed Boundary Integral Networks (PIBI-Nets): A Data-Driven Approach for Solving Partial Differential Equations. *Journal of Computational Science* 81 (Sept. 2024), 102355. https://doi.org/10.1016/j.jocs.2024.102355

[44] R.K. Nangia. 1992. Low-Speed Aerodynamics: from Wing Theory to Panel Methods J. Katz and A. Plotkin McGraw-Hill, McGraw Hill House, Shoppenhangers Road, Maidenhead, Berks, SL6 2QL. 1991. 632 pp. Illustrated. £32.95. *The Aeronautical Journal* 96, 955 (1992), 204–205. https://doi.org/10.1017/S0001924000024891

[45] Shaowu Pan, Steven L. Brunton, and J. Nathan Kutz. 2023. Neural Implicit Flow: A Mesh-Agnostic Dimensionality Reduction Paradigm of Spatio-Temporal Data. https://doi.org/10.48550/arXiv.2204.03216 arXiv:2204.03216 [cs]

[46] Blaine Quackenbush and PJ Atzberger. 2025. Transferable Foundation Models for Geometric Tasks on Point Cloud Representations: Geometric Neural Operators. *arXiv:2503.04649* (2025). https://arxiv.org/abs/2503.04649

[47] B Quackenbush and P J Atzberger. 2024. Geometric Neural Operators (Gnps) for Data-Driven Deep Learning in Non-Euclidean Settings. *Machine Learning: Science and Technology* 5, 4 (Dec. 2024), 045033. https://doi.org/10.1088/2632-2153/ad8980

[48] M. Raissi, P. Perdikaris, and G. E. Karniadakis. 2019. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comput. Phys.* 378 (Feb. 2019), 686–707. https://doi.org/10.1016/j.jcp.2018.10.045

[49] Stefan A Sauter and Christoph Schwab. 2010. Boundary element methods. In *Boundary Element Methods*. Springer, 183–287.

[50] Jia Sun, Yinghua Liu, Yizheng Wang, Zhenhan Yao, and Xiaoping Zheng. 2023. BINN: A Deep Learning Approach for Computational Mechanics Problems Based on Boundary Integral Equations. *Computer Methods in Applied Mechanics and Engineering* 410 (May 2023), 116012. https://doi.org/10.1016/j.cma.2023.116012 arXiv:2301.04480 [cs]

[51] Haixin Wang, Jiaxin Li, Anubhav Dwivedi, Kentaro Hara, and Tailin Wu. 2023. BENO: Boundary-embedded Neural Operators for Elliptic PDEs. In *NeurIPS 2023 AI for Science Workshop*.

[52] Colin White, Julius Berner, Jean Kossaifi, Mogab Elleithy, David Pitt, Daniel Leibovici, Zongyi Li, Kamyar Azizzadenesheli, and Anima Anandkumar. 2023. Physics-Informed Neural Operators with Exact Differentiation on Arbitrary Geometries. In *The Symbiosis of Deep Learning and Differential Equations III*.

[53] Minglang Yin, Nicolas Charon, Ryan Brody, Lu Lu, Natalia Trayanova, and Mauro Maggioni. 2024. DIMON: Learning Solution Operators of Partial Differential Equations on a Diffeomorphic Family of Domains. https://doi.org/10.48550/arXiv.2402.07250 arXiv:2402.07250 [cs]

[54] Chenyu Zeng, Yanshu Zhang, Jiayi Zhou, Yuhan Wang, Zilin Wang, Yuhao Liu, Lei Wu, and Daniel Zhengyu Huang. [n. d.]. Point Cloud Neural Operator for Parametric PDEs on Complex and Variable Geometries. *Computer Methods in Applied Mechanics and Engineering* 443 ([n. d.]), 118022. https://doi.org/10.1016/j.cma.2025.118022

A Related Methods

Neural Operators. Neural operators extend the neural network paradigm to learn mappings between infinite-dimensional functions [2, 4], making them particularly well-suited for approximating PDE solution operators. They have a universal approximation property for nonlinear operators [28], accept input functions on any discretization, and output functions that can be queried consistently at arbitrary resolutions. This flexibility improves data efficiency by supporting training with data of varying discretization and fidelity. Kossaifi et al. [27] maintain a comprehensive open-source library for learning neural operators in PyTorch.

Neural operator architectures such as the Spherical Fourier Neural Operator (SFNO) [6] and Optimal Transport Neural Operator (OTNO) [32] focus solely on the surface. While these approaches are computationally efficient, they cannot produce predictions over the full domain. In contrast, methods like the Geometry-Informed Neural Operator (GINO) [36] and Point-Cloud Neural Operator (PCNO) [54] can handle the boundary and the full domain together. However, since their computations are performed over the entire domain, their efficiency is reduced.

Boundary Element Methods By letting $\phi(y) = G(x, y)$. In equation (4), when $x \to \partial\Omega_\alpha$, we obtain

$$c(x)u(x) = \int_{\Omega_\alpha} G(x,y)f(y)\,dy + \int_{\partial\Omega_\alpha} \mathcal{B}(u, G, \alpha)(y)\,dS(y) \qquad \text{for } x \in \partial\Omega_\alpha, \tag{19}$$

where $c(x) = \frac{1}{2}$ if $\partial\Omega_\alpha$ is smooth. To solve the corresponding PDEs, boundary element methods (BEMs) [10, 49] reduce the dimensionality by only discretizing only the boundary rather than the entire domain. But they are also limited to the problem where an explicit form of fundamental solution is known.

Boundary Encoders. A straightforward approach to incorporating boundary information for solving PDEs is to use encoders that extract features from the boundary. Multiple architectures, such as GCN [26], MeshCNN [21], or pre-trained models [15, 46], can serve as the encoder. Various strategies can then be employed to integrate these boundary features. For example, attention-based mechanisms are used in [9, 12, 51], DeepONet architectures are employed in [16, 42]. However, these feature vectors often lose the underlying geometric structure, which limits their ability to generalize across varying geometries.

Physics-Informed Losses. Physics-informed loss functions can be used to solve PDEs, as demonstrated in physics-informed neural networks (PINNs) [48] and physics-informed neural operators (PINOs) [20, 37, 38]. Many variants of PINNs [19, 30, 43, 50] leverage physics-informed losses to enforce boundary conditions. More explicitly, letting $u_\theta := \mathcal{G}_\theta(f, g, \alpha)$, these approaches penalize deviations away from the PDE equations and boundary conditions via the PDE residual loss

$$\mathcal{L}_{\text{pde}} = \|\mathcal{P}u_\theta - f\|^2_{L^2(\Omega_\alpha)} + \lambda \|\mathcal{R}_\alpha u_\theta - g\|^2_{L^2(\Omega_\alpha)}. \tag{20}$$

or the boundary integral loss

$$\mathcal{L}_{\text{b}} = \left\| cu_\theta - \int_{\Omega_\alpha} G(x,y)f(y)\,dy - \int_{\partial\Omega_\alpha} \mathcal{B}(u_\theta, G, \alpha)(y)\,dS(y) \right\|^2_{L^2(\partial\Omega_\alpha)} \tag{21}$$

However, these physics-informed losses often require careful tuning of collocation points and can be sensitive to discretization, making training challenging for complex geometries or high-dimensional boundary value problems.

B Boundary Integral Equations

The differential operator for linear PDEs can be written as

$$\mathcal{P}u = \sum_{|k|\le m} a_k(x) \frac{\partial^{|k|} u}{\partial x_1^{k_1} \cdots \partial x_n^{k_n}} \tag{22}$$

Assume that the test function ϕ and a_k are smooth enough, for example, $\phi, a_k \in C^\infty(\overline{\Omega})$, by General Green Theorem (see equation (3.5.19) in [23]), we can derive the weak formulation of the original equation.

$$\int_{\Omega_\alpha} u\mathcal{P}^*\phi \, dx = \int_{\Omega_\alpha} f\phi \, dx + \int_{\partial\Omega_\alpha} \mathcal{B}(u,\phi,\alpha) \, dS, \tag{23}$$

where

$$\mathcal{P}^* : \phi \mapsto \sum_{|k|\le m} (-1)^{|k|} \frac{\partial^{|k|} a_k \phi}{\partial x_1^{k_1} \cdots \partial x_n^{k_n}} \tag{24}$$

is the adjoint operator of $\mathcal{P}$. $\mathcal{B}(u,\phi,\alpha)$ denotes boundary terms, $\mathcal{B}$ can be derived from repeatedly applying the divergence theorem

$$\begin{aligned}
\int_{\Omega_\alpha} \phi\mathcal{P}u &= \sum_{|k|\le m} \int_{\Omega_\alpha} a_k\phi \frac{\partial^{|k|} u}{\partial x_1^{k_1} \cdots \partial x_n^{k_n}} \, dx \\
&= \sum_{|k|\le m} - \int_{\Omega_\alpha} \frac{\partial a_k\phi}{\partial x_{j_1}} \frac{\partial^{|k|-1} u}{\partial x_1^{k_1} \cdots \partial x_{j_1}^{k_{j_1}-1} \cdots \partial x_n^{k_n}} \, dx \\
&\qquad + \int_{\partial\Omega_\alpha} a_k\phi \frac{\partial^{|k|-1} u}{\partial x_1^{k_1} \cdots \partial x_{j_1}^{k_{j_1}-1} \cdots \partial x_n^{k_n}} \nu_\alpha^{j_1} \, dS \\
&= \sum_{|k|\le m} \int_{\Omega_\alpha} \frac{\partial a_k\phi}{\partial x_{j_1}\partial x_{j_2}} \frac{\partial^{|k|-2} u}{\partial x_1^{k_1} \cdots \partial x_{j_1}^{k_{j_1}-1} \cdots \partial x_{j_2}^{k_{j_2}-1} \cdots \partial x_n^{k_n}} \, dx \\
&\qquad + \int_{\partial\Omega_\alpha} a_k\phi \frac{\partial^{|k|-1} u}{\partial x_1^{k_1} \cdots \partial x_{j_1}^{k_{j_1}-1} \cdots \partial x_n^{k_n}} \nu_\alpha^{j_1} \, dS \\
&\qquad + \int_{\partial\Omega_\alpha} \frac{\partial a_k\phi}{\partial x_{j_1}} \frac{\partial^{|k|-2} u}{\partial x_1^{k_1} \cdots \partial x_{j_1}^{k_{j_1}-1} \cdots \partial x_{j_2}^{k_{j_2}-1} \cdots \partial x_n^{k_n}} \nu_\alpha^{j_2} \, dS \\
&= \cdots
\end{aligned} \tag{25}$$

Therefore we can write the boundary term as $\mathcal{B}(u,\phi,\alpha) = \sum_j S_j(u)T_j(\phi)N_j(\nu_\alpha)$ which allows us to separate $T(\phi)$, as it does not depend on the shape parameter α.

For example, in the 2D Poisson equation,

$$\mathcal{B}(u,\phi,\alpha) = u\frac{\partial\phi}{\partial\nu_\alpha} - \phi\frac{\partial u}{\partial\nu_\alpha}.$$

Denoting

$$\nabla\phi = \begin{pmatrix}\phi_x\\ \phi_y\end{pmatrix}, \quad \nabla u = \begin{pmatrix}u_x\\ u_y\end{pmatrix}, \quad \partial\nu_\alpha = \begin{pmatrix}\nu_\alpha^x\\ \nu_\alpha^y\end{pmatrix},$$

we have

$$\mathcal{B}(u,\phi,\alpha) = u\phi_x\nu_\alpha^x + u\phi_y\nu_\alpha^y + u_x\phi\nu_\alpha^x + u_y\phi\nu_\alpha^y = (\phi_x,\phi_y,\phi,\phi)\cdot\begin{pmatrix}u\nu_\alpha^x\\ u\nu_\alpha^y\\ u_x\nu_\alpha^x\\ u_y\nu_\alpha^y\end{pmatrix}. \tag{26}$$

C Details about the Airfoil Experiment

C.1 Airfoil with Flap

The Airfoil-Flap dataset, originally generated in [54], is designed with the following far-field boundary conditions

$$\rho_\infty = 1.0, \quad p_\infty = 1.0, \quad M_\infty = 0.8. \tag{27}$$

Both the main airfoil and the flap are derived from NACA four-digit profiles. The flap positioned at $(-0.015, 0.05)$ relative to the main airfoil. The chord length of the main airfoil is 1.0 and the chord length of the flap is 0.2. Additional design parameters are summarized in Table 3.

Table 3: Geometric parameters for the shapes generated in Airfoil-Flap dataset (extracted from [54])

Design variable		Range
Main airfoil	Camber-to-chord ratio	$0\% \sim 9\%$
	Max camber location	$20\% \sim 60\%$
	Thickness-to-chord ratio	$5\% \sim 30\%$
	Angle of attack	$-5° \sim 20°$
Flap	Camber-to-chord ratio	$0\% \sim 9\%$
	Max camber location	$20\% \sim 60\%$
	Thickness-to-chord ratio	$10\% \sim 20\%$
	Relative angle of attack	$5° \sim 40°$

C.2 Some Results

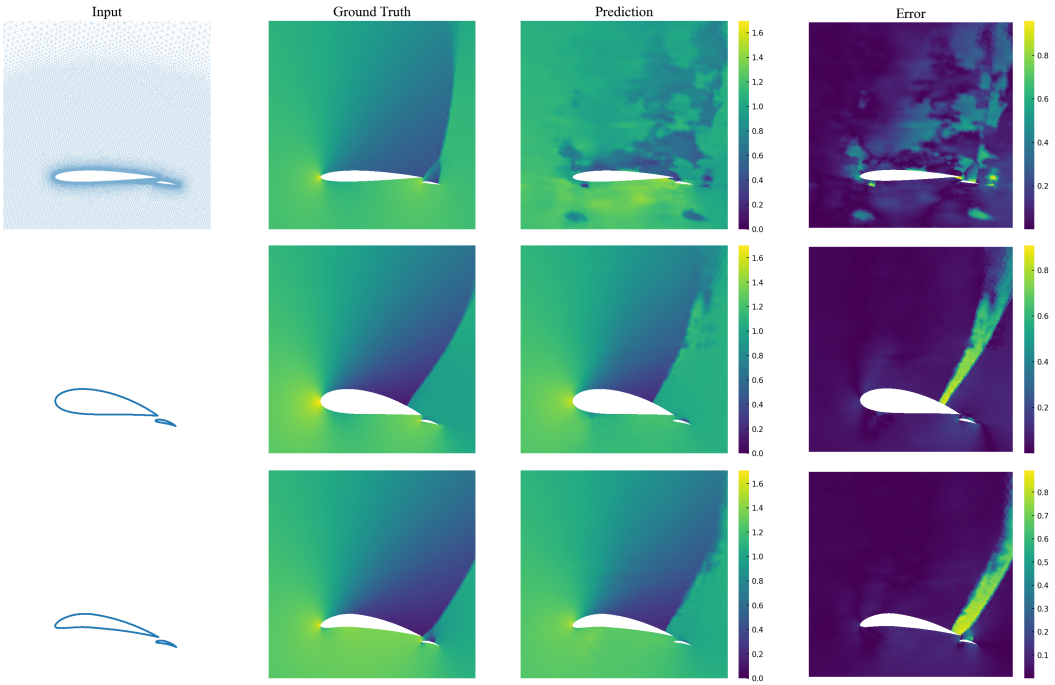

Figure 4: A comparison when the model is trained on *Airfoil* but tested on *Airfoil+Flap*. The top row shows the PCNO sample with the lowest loss, the middle row shows the ExtBNO sample with the median loss, and the bottom row shows the ExtBNO sample with the highest loss.

D Details about the Laplace Equation

D.1 Laplace Equation

The Laplace equation with Dirichlet boundary condition can be expressed as

$$\begin{aligned} \Delta u &= 0, \\ u|_{\partial\Omega} &= g. \end{aligned} \tag{28}$$

To investigate generalization capabilities, we consider a variety of geometric configurations by sampling domain shapes from a Gaussian process defined over polar coordinates. Specifically, the boundary $\partial\Omega$ of each domain is parameterized as

$$(x(\theta), y(\theta)) = (r(\theta)\cos\theta, r(\theta)\sin\theta) \tag{29}$$

where the radial function $r(\theta)$ is sampled from a Gaussian process.

Let $r \sim \mathcal{GP}(0, C(\theta, \theta'))$, with covariance kernel

$$C(\theta, \theta') = \frac{1}{2}(\theta - \theta')^2 - \pi(\theta - \theta') + \frac{1}{3}\pi^2, \quad \theta - \theta' \in [0, 2\pi] \tag{30}$$

whose Fourier eigenvalues decay as $\lambda_k = \frac{1}{k^2}$. This yields the following Karhunen-Loève expansion

$$r(\theta) = \sum_{k \in \mathbb{Z}\setminus\{0\}} \frac{1}{k} z_k e^{ik\theta}, \quad z_k \sim \mathcal{CN}(0, 1) \ i.i.d. \tag{31}$$

For numerical implementation, the series is truncated at frequency K, resulting in

$$r_K(\theta) = \sum_{-K \le k \le K, k \neq 0} \frac{1}{k} z_k e^{ik\theta}, \quad z_k \sim \mathcal{CN}(0, 1) \ i.i.d. \tag{32}$$

To ensure stability and prevent degeneracy near the origin, we normalize the radial function as

$$\tilde{r}_K = 1 + \frac{r_K}{S_K}, \quad S_K = 2 \sum_{1 \le k \le K} \frac{1}{k} \tag{33}$$

and the resulting normalized process is denoted by Gaussian process is denoted as $\mathcal{GP}_K$.

The following four types of domain geometries are considered:

- **Low Frequency:** A single simply connected domain generated by $r \sim 1 + \mathcal{GP}_{K=10}$, resulting in a smooth boundary with low-frequency features.
- **High Frequency:** A single simply connected domain generated by $r \sim 1 + \mathcal{GP}_{K=150}$, leading to a highly oscillatory boundary with rich high-frequency features.
- **Double:** A single connected domain formed by joining two independently sampled boundaries $r_1, r_2 \sim 1 + \mathcal{GP}_{K=10}$. r_2 is then resclaed by $\alpha \sim \mathcal{U}(0.5, 1)$. The two components are positioned to avoid overlap and are smoothly connected at the origin using a spline-based interpolation.
- **Hole:** A domain consisting of an outer and an inner boundary. The outer boundary is sampled from $r \sim 1 + \mathcal{GP}_{K=10}$, as in the simply connected case. A random scaling ratio β is then sampled form $\beta \sim 0.4 + 0.1\mathcal{GP}_{K=5}$ to define the inner boundary as a scaled version of the outer boundary, centered within it.

In addition to varying domain geometries, g is drawn from another Gaussian process defined over the spatial domain, with a radial basis function (RBF) kernel given by

$$C(x, y) = e^{-\frac{|x-y|^2}{2L^2}} \tag{34}$$

where $L > 0$ denotes the length scale parameter that controls the smoothness of the samples. For numerical implementation, periodic boundary conditions are imposed, allowing us to represent the process using a truncated KL expansion. The corresponding eigenvalues are given by

$$\lambda_k = 2\pi L^2 e^{-\frac{1}{2}\left|\frac{2\pi k}{T}\right|^2 L^2} \tag{35}$$

and the truncated expansion takes the form

$$u_L(x) = \sum_{-K \le k_x, k_y \le K} \sqrt{\lambda_k} e^{\frac{2\pi i k \cdot x}{T}} \tag{36}$$

where $K = 10, T = 4$.

To ensure comparability across different length scales, each sample is normalized as

$$\tilde{u}_L = \frac{u_L}{\sigma_L 2\pi L^2} \tag{37}$$

where σ_L is a scaling factor chosen such that $\mathbb{E}[\max \tilde{u}_L] - \mathbb{E}[\min \tilde{u}_L] = 1$ and is approximated using 200 samples of the process. In our setup, the length scale of g is fixed at $L_g = \frac{1}{10}\sqrt{2}$.

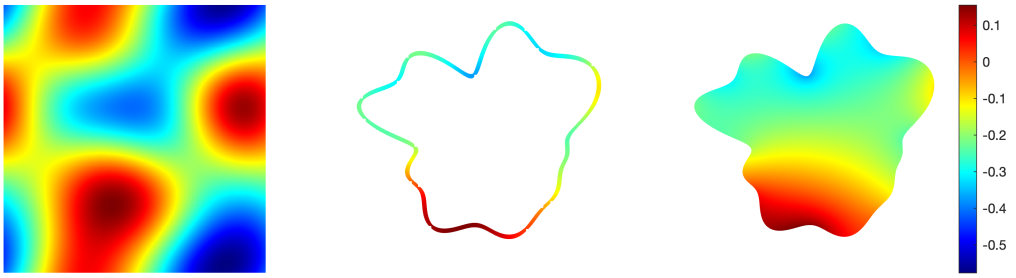

Figure 5: Illustration of the case with $L_g = \frac{1}{10}\sqrt{2}$. From left to right: realizations of the Gaussian random field, corresponding boundary values and the Laplace equation solutions.

D.2 Additional Results Figures

The following figures present the results obtained under different training shape configurations. For each setting, we visualize the test sample that achieves the median L_2 error.

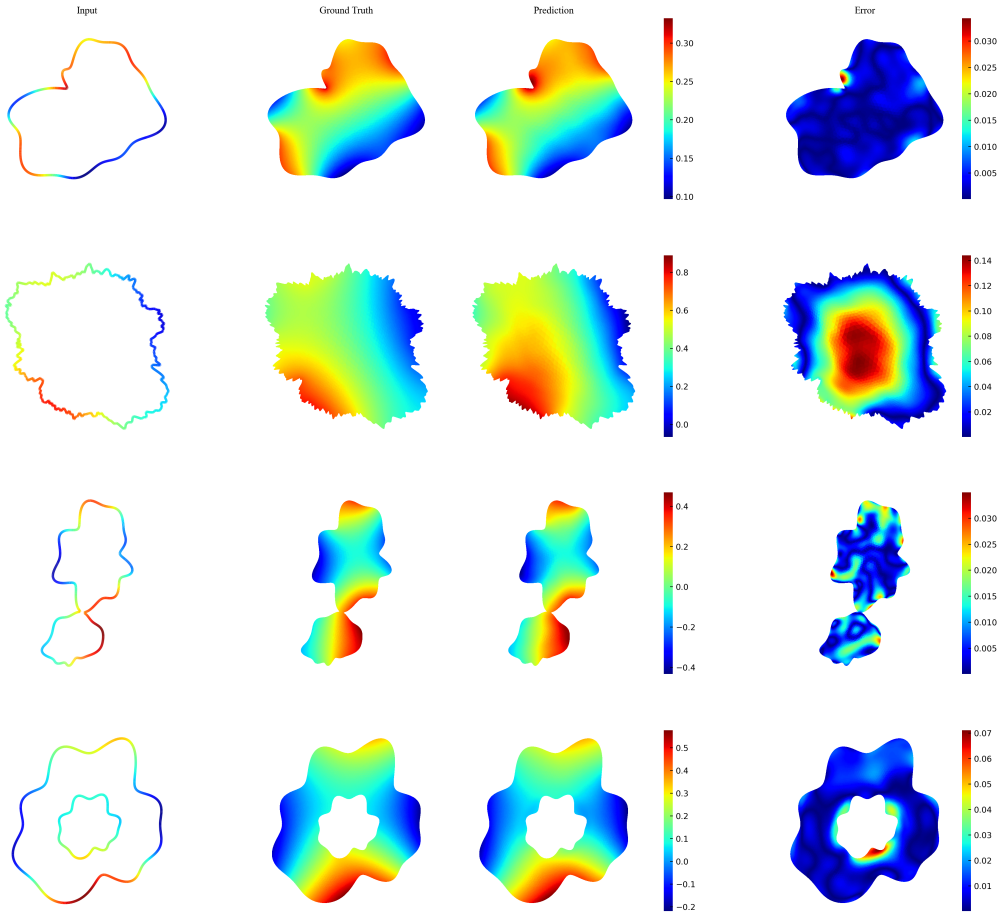

Figure 6: Model trained solely on *Low-Freq* shapes and evaluated on all other shape families. The visualized cases are those attaining the median L_2 error.

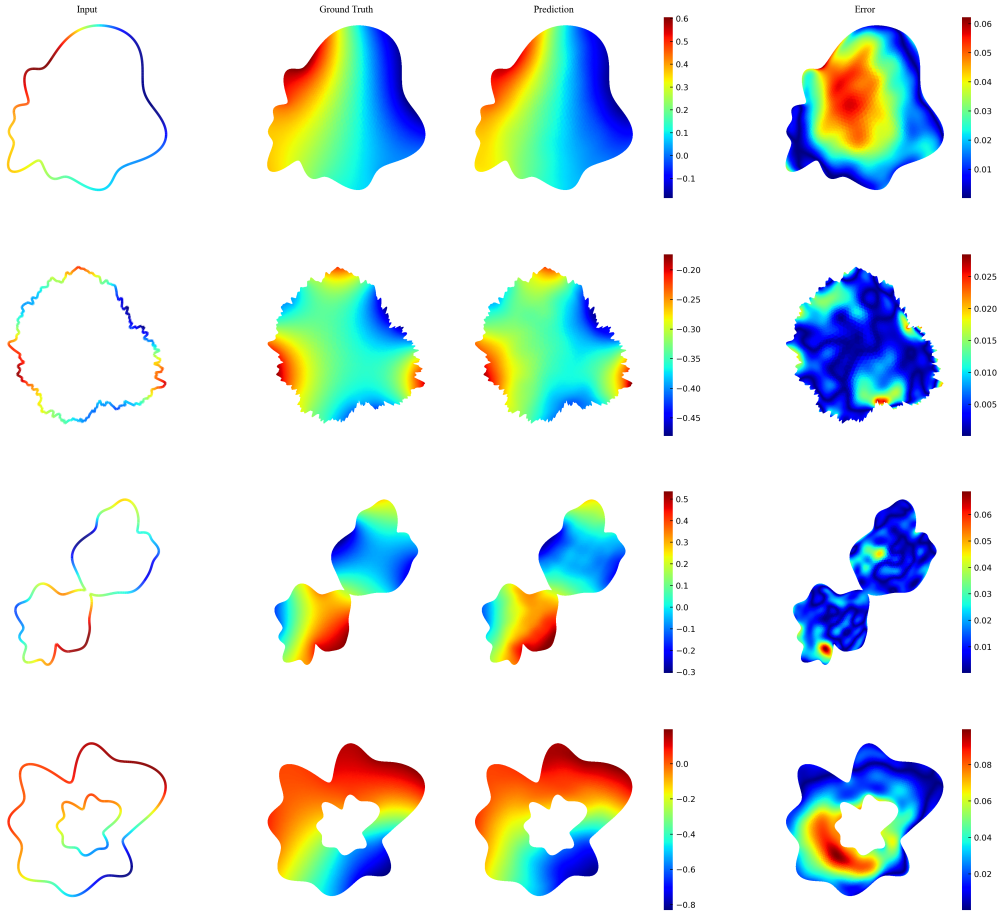

Figure 7: Model trained solely on *High-Freq* shapes and evaluated on all other shape families. The visualized cases are those attaining the median L_2 error.

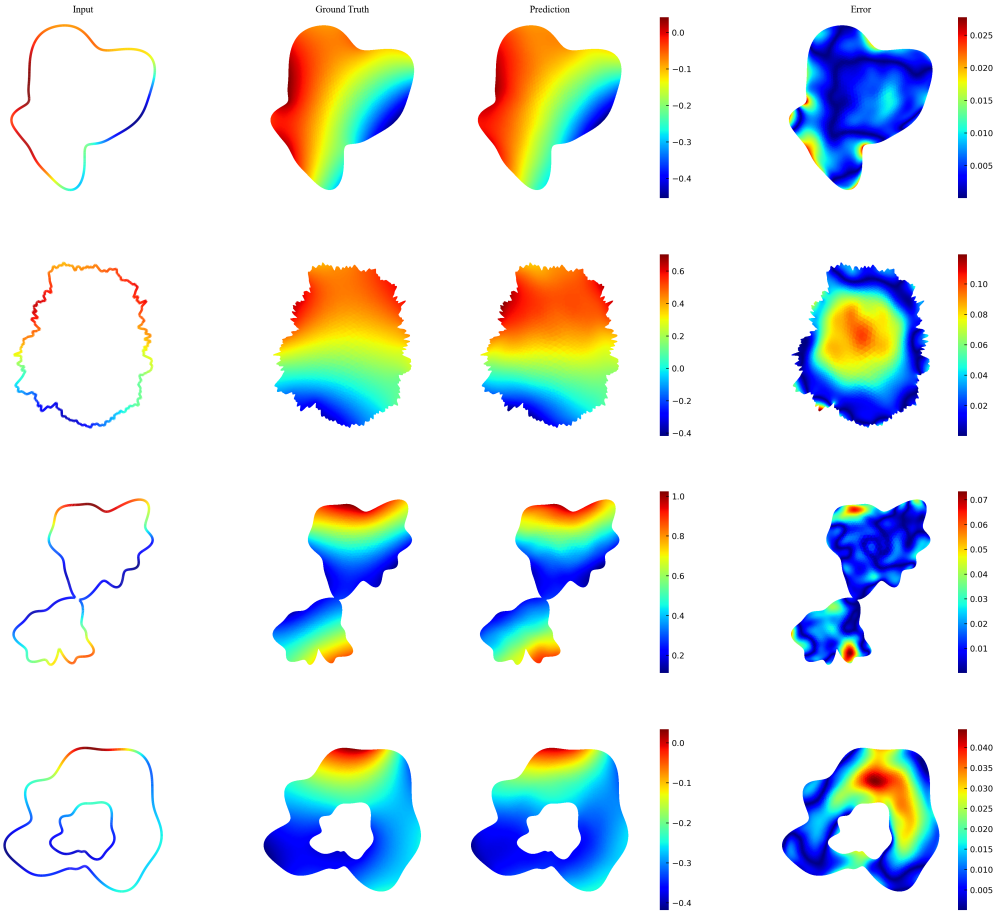

Figure 8: Model trained solely on *Double* shapes and evaluated on all other shape families. The visualized cases are those attaining the median L_2 error.

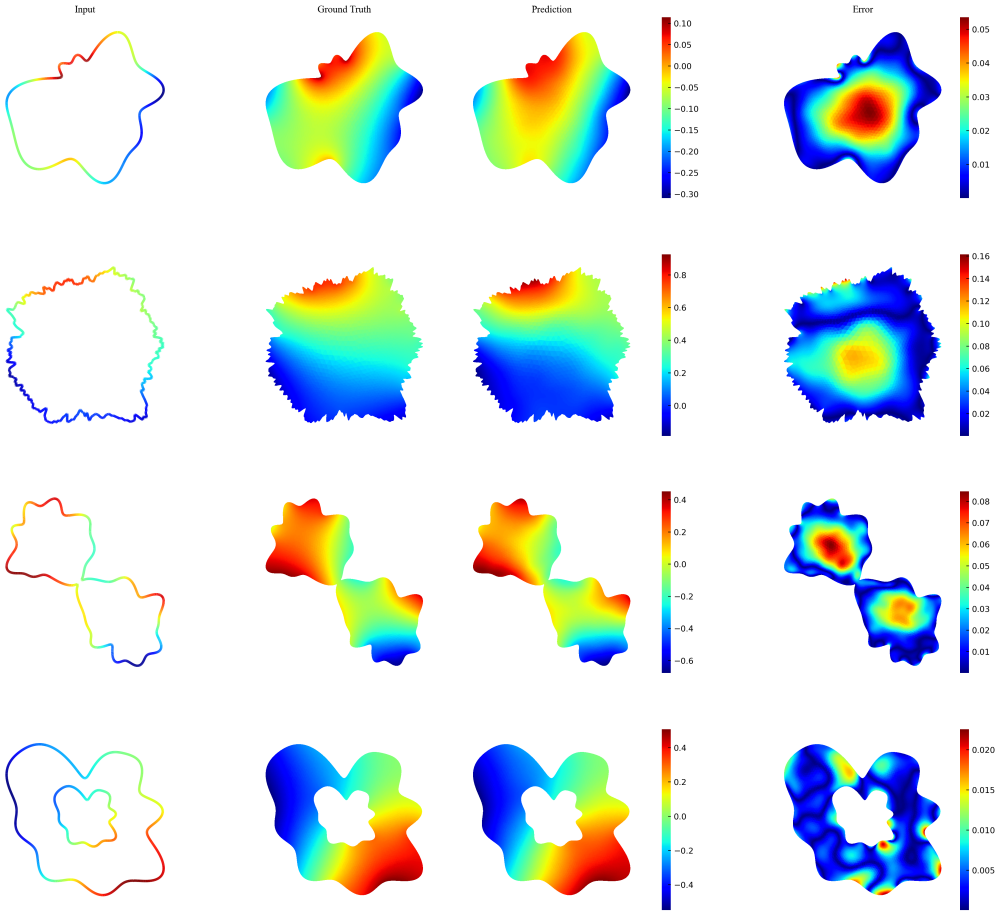

Figure 9: Model trained solely on *Double* shapes and evaluated on all other shape families. The visualized cases are those attaining the median L_2 error.

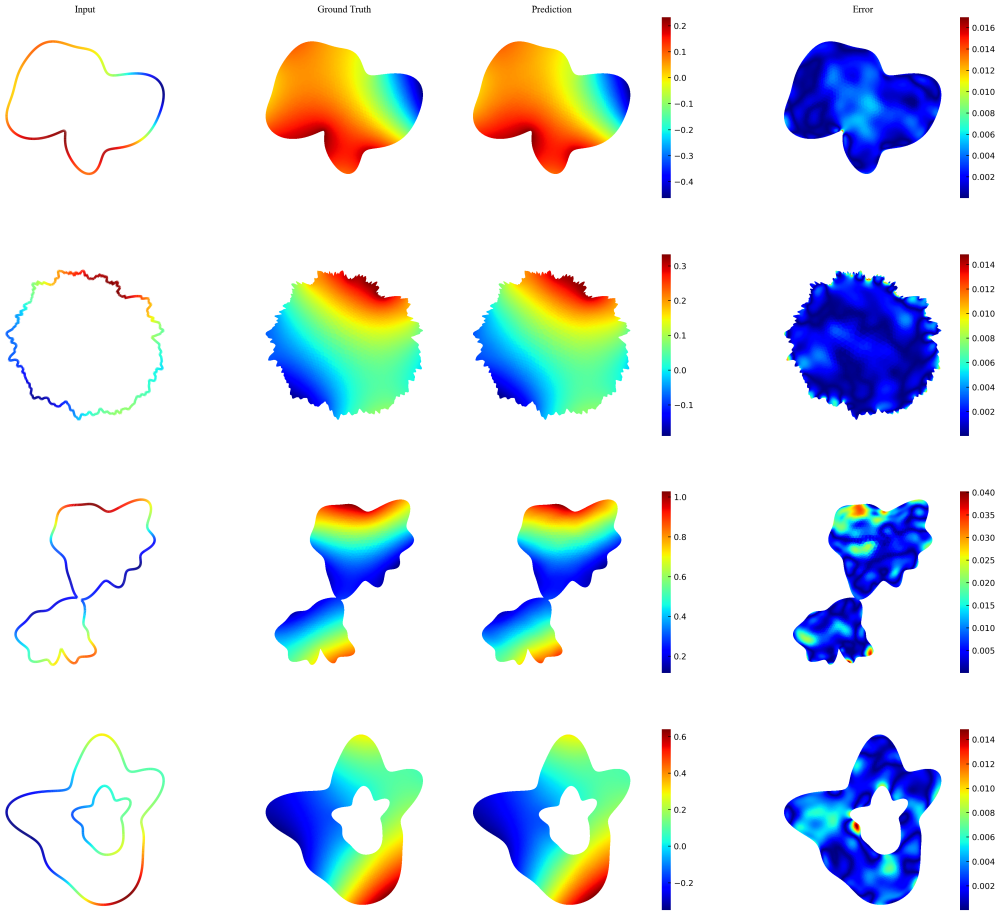

Figure 10: Model trained on 128 samples from each shape type (512 samples in total). The visualized cases are those attaining the median L_2 error.