A COMPREHENSIVE COMPARISON OF NEURAL OPERATORS FOR 3D INDUSTRY-SCALE ENGINEERING DESIGNS

A PREPRINT

Weiheng Zhong[1]* Qibang Liu[2] Diab Abueidda[2] Seid Koric[2,3] Hadi Meidani[1]

[1] Department of Civil and Environmental Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801
[2] National Center for Supercomputing Applications, University of Illinois Urbana-Champaign, Urbana, IL 61801
[3] Department of Mechanical Science and Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801

ABSTRACT

Neural operators have emerged as powerful tools for learning nonlinear mappings between function spaces, enabling real-time prediction of complex dynamics in diverse scientific and engineering applications. With their growing adoption in engineering design evaluation, a wide range of neural operator architectures have been proposed for various problem settings. However, model selection remains challenging due to the absence of fair and comprehensive comparisons. To address this, we propose and standardize six representative 3D industry-scale engineering design datasets spanning thermal analysis, linear elasticity, elasto-plasticity, time-dependent plastic problems, and computational fluid dynamics. All datasets include fully preprocessed inputs and outputs for model training, making them directly usable across diverse neural operator architectures. Using these datasets, we conduct a systematic comparison of four types of neural operator variants, including Branch-Trunk-based Neural Operators inspired by DeepONet, Graph-based Neural Operators inspired by Graph Neural Networks, Grid-based Neural Operators inspired by Fourier Neural Operators, and Point-based Neural Operators inspired by PointNet. We further introduce practical enhancements to adapt these models to different engineering settings, improving the fairness of the comparison. Our benchmarking study evaluates each model's strengths and limitations in terms of predictive performance, computational efficiency, memory usage, and deployment complexity. The findings provide actionable insights to guide future neural operator development. All **source code** is available to the public at Github. The **data** is available at Harvard Dataverse for the paper reviewing process: Heat sink, Bracket, Bracket-time, JEB, and DrivAer and DriverNet++

Keywords Neural Operator, Fair Comparison, 3D Engineering Design, Industry-Scale Problems

1 Introduction

The numerical solution of partial differential equations (PDEs) is a foundational aspect of computational science and engineering, which can be used in instant evaluations in design, sensitivity analysis and uncertainty quantification, online controls, and digital twin development. Traditional discretization-based methods, including the finite element method (FEM) Jagota et al. [2013] and finite volume method (FVM) Eymard et al. [2000], have been extensively developed, offering high accuracy and robustness across various domains. However, despite their widespread use, these conventional approaches often incur substantial computational costs, especially for large-scale, high-fidelity simulations. In recent years, neural operators have emerged as a promising alternative, capable of learning mappings between infinite-dimensional function spaces directly from data Li et al. [2020b], Lu et al. [2021b]. By bypassing

*Corresponding author: weiheng4@illinois.edu

traditional mesh discretization at inference time, neural operators significantly accelerate PDE solution prediction, unlocking new possibilities for real-time simulation, optimization, and control Azizzadenesheli et al. [2024].

One of the most critical applications of PDE solvers lies in engineering design evaluation, where rapid assessments of physical fields (e.g., temperature, stress, pressure) are essential for exploring a wide range of candidate designs Azizzadenesheli et al. [2024]. In this context, neural operators offer the potential to significantly accelerate design iteration cycles. While many powerful neural operator architectures have been proposed, existing evaluation frameworks have not been rigorously tested on industry-scale applications that demand both complex 3D geometries with high-resolution meshes (typically exceeding 10,000 nodes) and parametric representations—two fundamental characteristics of real-world engineering workflows. High-resolution 3D meshes are essential for accurately capturing the fine geometric details required in structural, thermal, and fluid simulations. Likewise, parametric representations are critical for manufacturability, as they enable design control through a small number of interpretable parameters such as length, thickness, or curvature Feng et al. [2019], He et al. [2024a], Lu et al. [2022b], Patterson and Allison [2022]. However, most current benchmarks fall short of these requirements. First, the majority of datasets used to evaluate neural operators are predominantly 2D Hao et al. [2023], Wu et al. [2024, 2023], Li and Ye [2025], Yu et al. [2024a], Serrano et al. [2023b], Liu et al. [2023b], Serrano et al. [2024], limiting their relevance for applications where full 3D modeling is essential. Second, existing evaluations often rely on free-form geometries Li et al. [2023c], Wu et al. [2024], Hao et al. [2023], which are rarely adopted in practical engineering due to manufacturing constraints. These mismatches between current evaluation practices and real-world design needs reveal a significant gap, underscoring the importance of developing more representative benchmarking frameworks for neural operators in industry-scale applications.

To bridge this gap, we introduce a standardized benchmarking framework for neural operators in 3D, parameterized, industry-scale engineering problems. Our approach begins with curating six diverse datasets that encompass a wide range of engineering challenges, including thermal analysis, linear elasticity, elasto-plastic deformation, time-dependent plasticity, and computational fluid dynamics. These datasets are designed to capture the complexity of real-world applications. We then systematically classify state-of-the-art neural operator architectures into four major categories—Branch-Trunk-based Lu et al. [2021b], He et al. [2024b,a], Zhong and Meidani [2024, 2025], Graph-based Li et al. [2020c], Gladstone et al. [2024], Grid-based Li et al. [2023c], Choy et al. [2025], and Point-based operators Hao et al. [2023], Kashefi and Mukerji [2022], Wu et al. [2024]. From each category, we identify representative models and ensure their fair adaptation and evaluation on all datasets. Our benchmarking process comprehensively assesses these models, focusing on predictive accuracy, computational efficiency, memory usage, and deployment practicality. Our contributions are summarized as follows:

- We curated six industry-level 3D engineering datasets, enabling a wide range of neural operators training and evaluation.
- We present a comprehensive taxonomy of neural operator architectures, adapting each model to both parametric and free-form geometries and proposing architectural enhancements to ensure fair comparisons.
- We conduct the first extensive benchmarking study of neural operators on diverse industry-scale engineering designs, providing actionable insights for future neural operator development and deployment.
- We release all datasets, model implementations, and evaluation code, promoting reproducibility and facilitating further research in neural operators for industry-scale engineering design.

2 Related works

2.1 DeepONet-inspired Neural Operator

Deep Operator Networks (DeepONets), introduced by Lu et al. [2021b]., are neural architectures designed to learn nonlinear operators—mappings between infinite-dimensional function spaces—enabling efficient approximation of complex physical models from limited data Lu et al. [2021a, 2019]. Their core structure consists of a branch network encoding input functions at fixed sensor locations and a trunk network processing output coordinates, with outputs combined via a dot product Lu et al. [2021a]. Supported by a universal operator approximation theorem, DeepONet generalizes well to nonlinear and parametric PDEs, while physics-informed variants incorporate governing equations directly into the loss to avoid numerical solvers Lu et al. [2021a], Yang [2024].

To address scalability and uncertainty, Separable DeepONet uses factorized subnetworks to mitigate the curse of dimensionality Mandl et al. [2024], and α-VI DeepONet adopts variational inference for robust uncertainty quantification Wang [2024b]. Computational efficiency is further improved via RaNN-DeepONet's randomized networks Zhang et al. [2025], derivative-enhanced losses Zhang [2024], and Point-DeepONet's mesh-free learning on 3D point clouds Park and Kang [2025].

Applications span porous media flow, carbon sequestration, and PDE-constrained control. Architectures like U-DeepONet employ U-Net backbones to improve multiphase flow modeling Diab and Al Kobaisi [2024], Li [2024], while nonlinear reconstructions extend DeepONet to PDEs with discontinuities Lanthaler et al. [2022]. Transfer learning techniques enhance adaptability across problem settings, enabling efficient fine-tuning for multi-operator tasks Goswami et al. [2022], Yadav [2023]. Overall, DeepONet remains a powerful and evolving tool for physics-informed machine learning.

2.2 Graph-based Neural Operator

Graph Neural Operators (GNOs) have emerged as powerful tools for learning mappings between function spaces defined on irregular domains, particularly for solving partial differential equations (PDEs). A pivotal example is the Multipole Graph Neural Operator (MGNO) Li et al. [2020d], which incorporates a hierarchical, multi-level graph representation to capture long-range interactions in PDE solutions efficiently. This approach reduces computational complexity from quadratic to linear in the number of nodes, thereby enabling scalable and discretization-invariant GNO architectures. Building upon the GNO framework, the Latent Neural Operator (LNO) Wang [2024a] encodes inputs into a latent graph space enriched by a Physics-Cross-Attention mechanism to learn more expressive graph-based operator mappings. CORAL Serrano et al. [2023a] extends GNOs by using coordinate-based neural representations, allowing operators to generalize to PDEs defined on arbitrary and non-mesh-conforming geometries. The Geometry-Informed Neural Operator (GINO) Li et al. [2023c] further integrates graph-based and spectral operator learning to construct latent regular grids from unstructured 3D domains for solving large-scale PDEs. PDE-GCN Eliasof et al. [2021] exemplifies a graph neural operator architecture derived from PDE discretizations, which combats over-smoothing in deep GNN layers through PDE-aware message passing. To improve numerical stability, the Representation Equivalent Neural Operator (RENO) framework Serrano [2023] introduces alias-free GNOs by aligning discrete GNN computations with the properties of continuous operators. Scalable Transformer variants adapted for graph operator learning Wu et al. [2023] allow GNOs to handle high-dimensional PDE surrogate modeling tasks while maintaining strong accuracy and efficiency. Graph-Tuple Neural Networks (GtNNs) Velasco et al. [2024] expand GNOs to multimodal node and edge features with theoretical guarantees for stability under perturbations. Lastly, the Alias-Free Mamba Neural Operator (MambaNO) Zheng et al. [2024a] achieves linear-complexity GNO inference by combining long-range dependency modeling and implicit time integration, offering a new paradigm for graph-based operator learning.

2.3 Grid-based Neural Operator

Fourier Neural Operators (FNOs) represent a groundbreaking approach in neural operator learning, efficiently approximating solution operators of parametric partial differential equations (PDEs) by operating in the frequency domain Li et al. [2020a]. They utilize Fourier transforms with learned spectral filters to capture global dependencies and achieve resolution-invariant learning with quasi-linear time complexity, excelling in modeling complex systems such as turbulent flows and fluid dynamics Li et al. [2020a, 2021]. The architecture lifts input functions to higher-dimensional representations, applies Fourier domain operations, and reconstructs target functions, providing high accuracy and computational efficiency compared to classical solvers Li et al. [2020a].

Extensions enhancing FNO scalability, flexibility, and applicability include multi-grid architectures like MgFNO and Multi-Grid Tensorized FNO (MG-TFNO), which leverage hierarchical frequency decompositions to improve generalization and reduce memory usage for large-scale PDE problems Kossaifi et al. [2023], Li and Ye [2025]. Domain Agnostic FNOs (DAFNO) address irregular grids and diverse domain topologies, while learnable Fourier filters enable more localized spectral feature extraction Liu et al. [2023a], Qi and Sun [2024]. Hybrid models integrating FNOs with CNNs or graph neural operators combine local spatial and global spectral information for enhanced predictive performance Liu-Schiaffini et al. [2024]. Variants such as U-FNO and HyperFNO demonstrate superior speed, accuracy, and data efficiency in multiphase flow and PDE simulations Wen et al. [2022].

Theoretical analyses confirm FNOs' universal approximation capabilities for continuous operators and convergence under discretization refinements Kovachki et al. [2021]. Physics-informed training regimes incorporate PDE constraints directly into learning to improve accuracy and stability Li et al. [2024]. Extensive benchmarks demonstrate FNOs outperform classical numerical solvers with speed-ups exceeding 100×, while retaining high fidelity in fluid dynamics, Darcy flows, and turbulent simulations Lu et al. [2022a]. FNOs and their innovations thus form a versatile and efficient framework pivotal to advancing scientific machine learning.

2.4 Point-based Neural Operator

Point-based neural operator models have emerged as a powerful approach for learning solution operators of partial differential equations (PDEs) by directly operating on spatial point clouds. This representation enables flexibility in handling irregular geometries, complex boundary conditions, and variable discretization. The Geometry-Informed Neural Operator (GINO) introduced by Li et al. Li et al. [2023b] exemplifies this approach by combining graph neural networks on point clouds with Fourier Neural Operators acting on latent regular grids derived from geometric encodings such as signed distance functions. GINO achieves remarkable computational acceleration and discretization invariance, enabling efficient and accurate surrogate modeling for large-scale 3D PDEs in industrial scenarios.

Transformer-based architectures further advance point-based modeling of PDEs. Li et al. [2023a] and Wu et al. [2024] propose scalable pointwise transformers—such as Transolver—that utilize attention mechanisms over spatial points, effectively learning solution operators on general geometries with arbitrary discretizations. The Universal Physics Transformers (UPTs) framework Alkin et al. [2024] unifies spatio-temporal PDE modeling with shared transformer architectures acting on point-level data, demonstrating versatility and computational efficiency Alkin et al. [2024], Yu et al. [2024b]. Complementarily, coordinate-based neural field methods like CORAL Serrano et al. [2023c] bypass traditional meshing by learning operator mappings directly from point samples, preserving spatial structure and handling complex geometries robustly.

Recent innovations include alias-free and nonlocal operator models that use pointwise attention to uncover interpretable physics and mitigate numerical artifacts, as seen in the Alias-Free Mamba Neural Operator Zheng et al. [2024b] and the Nonlocal Attention Operator Yu et al. [2024b]. Graph-based models such as GFN Morrison et al. [2024] and inducing point transformer methods Lee and Oh [2024] also operate on point discretizations to ensure resolution invariance and scalability. Enhanced operator learning incorporates derivative information Qiu et al. [2024] and Bayesian uncertainty quantification, improving accuracy and robustness. Collectively, these point-based neural operators provide flexible, scalable, and accurate methods for PDE surrogate modeling on complex domains, pushing forward the frontiers of computational scientific modeling.

3 Evaluation framework on industry-scale engineering problems

A neural operator learns a mapping between infinite-dimensional function spaces. Let $D \subset \mathbb{R}^d$ be a Lipschitz domain Cobzaş et al. [2019], and let $\mathcal{B}$ be a Banach space of real-valued functions defined on D. Consider a family of level set functions Osher and Fedkiw [2001] $\mathcal{S} \subset \mathcal{B}$, where each $s \in \mathcal{S}$ defines a d-dimensional submanifold $\Omega_S = \{x \in D : s(x) > 0\}$. Given a partial differential operator $\mathcal{L}$, we study the solution $u \in \mathcal{U}_T \subset \mathcal{B}$ of the following PDE:

$$\mathcal{L}(u(x)) = f(x), \quad x \in \Omega_S, \tag{1}$$

$$u(x) = g(x), \quad x \in \partial\Omega_S, \tag{2}$$

where $f \in \mathcal{F} \subset \mathcal{B}$ denotes the PDE forcing term, $g \in \mathcal{G} \subset \mathcal{B}$ represents boundary and initial conditions, and $x \in D$ is the spatial coordinate. The neural operator seeks to approximate the solution operator as a mapping from PDE parameters and domain geometries to PDE solutions:

$$\mathcal{O} : \mathcal{F} \times \mathcal{G} \times \mathcal{S} \to \mathcal{U}, \tag{3}$$

where $\mathcal{U}$ is a Banach space representing the family of PDE solutions on varying domains Ω_S. In practice, neural operator models are trained on a finite collection of discrete observations of these functions Li et al. [2020b]. In many engineering optimization problems, parametric representations of PDE parameters (e.g., loading, inlet velocity) and design geometries are available. Accordingly, we predict the PDE solution with a neural operator model M in the

following setting:

$$u(x_q) = M(x_q, F, G, S, p), \tag{4}$$

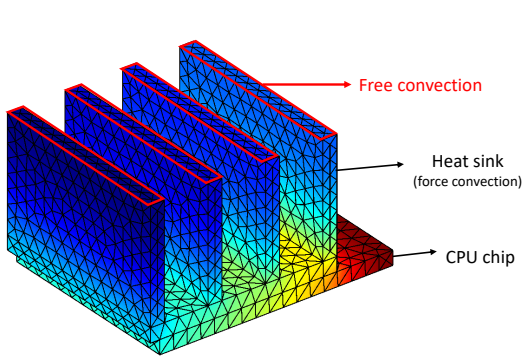
where $x_q \in D$ is a query point at which the PDE solution is predicted, $F = \{(x_i, f(x_i))\}_{i=1}^m$, $G = \{(x_i, g(x_i))\}_{i=1}^n$, and $S = \{x_i\}_{i=1}^l$ represent the point cloud representations of the parameter f, the boundary/initial conditions g, and the geometry Ω_S, respectively, and the vector $p \in \mathbb{R}^q$ denotes a q-dimensional parametric representation of the design space. We evaluate model performance across six real-world engineering datasets, covering a diverse range of physics and geometry representations:

- **Heatsink:** A thermal conduction dataset simulating heat dissipation from a CPU via a heat sink. The geometry is parameterized by fin height, fin thickness, fin spacing, and the number of fins. Boundary conditions include force convection on the side surfaces and free convection on the top. The steady-state temperature field is computed as the PDE solution.
- **Bracket:** This dataset models an elasto-plastic deformation problem in mechanical brackets with parameterized geometry, including length, thickness, and hole radius. Each sample also varies in applied pressure loading. FEM simulations use quadratic hexahedral elements, an elastic-plastic constitutive law with isotropic hardening, and small deformation assumptions.
- **Bracket-time:** A time-dependent extension of the bracket dataset, where varying loading sequences are applied over 101 time steps. The goal is to learn the final stress state of the bracket in elasto-plastic modeling.
- **JEB:** The Jet Engine Bracket (JEB) dataset consists of 2,138 geometrically diverse 3D bracket designs sourced from the GE Jet Engine Bracket Challenge Hong et al. [2025]. The shapes exhibit significant freeform variation without a shared parameterization. The task is to predict the von Mises stress field under a prescribed vertical load.
- **DrivAer:** A large-scale CFD dataset offering high-fidelity aerodynamic simulations across multiple 3D vehicle geometries with parametric variation Elrefaie et al. [2024a]. The objective is to predict surface pressure distribution under standardized wind flow conditions.
- **DrivAer++:** An extension of DrivAerNet involving freeform, non-parametric vehicle geometries Elrefaie et al. [2024b]. This dataset focuses on evaluating model generalization to unparameterized, complex CAD geometries in automotive fluid dynamics.

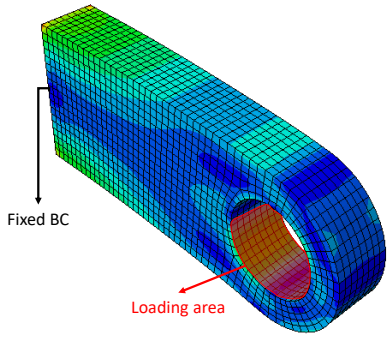
In summary, each dataset is a representation of one type of engineering problem, which is summarized in Table 1 The Visualization of the sample geometry and PDE solutions are shown in 1.

Table 1: Summary of datasets with graph size and associated challenges.

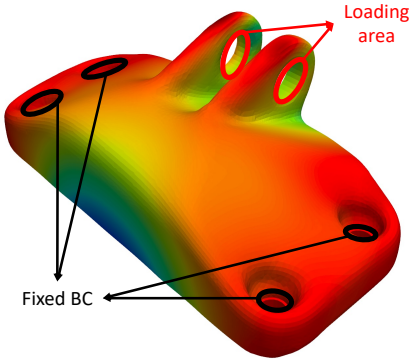
Name	Type	Data Size	Mesh Size	Geometry Representation	Challenges
Heat sink	Thermal	625	∼10k	Parametric	Varying design geometry
Bracket	Plastic mechanics	3,000	∼10k	Parametric	Varying geometry and loading
Bracket-dynamic	Plastic mechanics	10,000	∼30k	N/A (Fixed geometry)	Time-dependent loading
JEB	Elastic mechanics	2,000	∼100k	Non-parametric	Small set of complex geometry samples
DrivAerNet	CFD	4,000	∼500k	Parametric	Large mesh
DrivAerNet++	CFD	4,000	∼400k	Non-parametric	Large mesh with non-parametric geometry



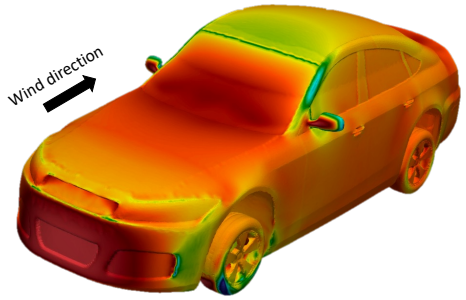
(a) Heat sink simulation



(b) Bracket simulation



(c) JEB dataset



(d) CFD simulation

Figure 1: Branch-trunk model architecture enhancements for freeform geometry.

4 Neural operator architectures

4.1 Branch-Trunk-Based Neural Operators

As one of the earliest neural operator architectures, Deep Operator Networks (DeepONet) Lu et al. [2021b] leverage the universal approximation theorem for operators by representing the PDE solution as an inner product between two neural networks: the branch network and the trunk network. The predicted solution at a query location x_q is given by:

$$u(x_q) = \text{Sum}(b \odot t) = \text{Sum}(\mathcal{U}_b(p) \odot \mathcal{U}_t(x_q)), \tag{5}$$

where $\mathcal{U}_b$ is a multi-layer perceptron (MLP) applied to the input parameter p, $\mathcal{U}_t$ is an MLP applied to the spatial query x_q, and $\odot$ represents the element-wise multiplications. However, the simplicity of this architecture limits its effectiveness in addressing complex engineering problems. For model performance improvement, Geom-DeepONet He et al. [2024a] replaces the branch MLP with a sinusoidal representation network (SIREN). On the other side, the Deep Compositional Operator Network (DCON) Zhong and Meidani [2024] increases model expressiveness by introducing a hierarchical composition of operator layers, For example, with two layers, the solution is approximated by:

$$u(x_q) = \mathcal{U}_u^2 \left(b \odot \mathcal{U}_u^1 (b \odot t)\right), \tag{6}$$

where $\mathcal{U}_u^1$ and $\mathcal{U}_u^2$ are learnable MLPs. As for problems with time-dependent PDEs, S-DeepONet He et al. [2024b] replaces the branch MLP with a gated recurrent unit (GRU), enabling temporal sequence encoding in the branch net:

$$b = \text{GRU}([f(t_0), \ldots, f(t_m)]), \tag{7}$$

where $f(t_k)$ denotes a time-dependent scalar function values at time step t_k. To enable handling of free-form geometries, Geometry-aware Neural Operator (GANO) Zhong and Meidani [2025] extends DCON by incorporating point-cloud-based geometric encoding, where a global feature vector E_g is concatenated with the trunk input:

$$u(x_q) = \mathcal{U}_u^2 \left(b \odot \mathcal{U}_u^1 (b \odot [t \parallel E_g]) \right), \tag{8}$$

$$E_g = \text{Pooling}(\{\mathcal{U}_g(x_i)\}_{i=1}^l), \tag{9}$$

where $\parallel$ denotes concatenation, and $\mathcal{U}_g$ encodes each geometry point x_i. Overall, compared with traditional DeepONet, these improved branch-trunk models provide complementary strengths for different engineering problems.

4.2 Graph-Based Neural Operators

Graph Neural Operators (GNOs) Li et al. [2020c] approximate mappings between function spaces using message passing over graphs, making them suitable for solving PDEs on irregular meshes. A typical GNO predicts the PDE solution on the mesh point coordinate x_i with stacked graph convolutions:

$$u(x_i) = \text{GC}^l \circ \text{GC}^{l-1} ... \circ \text{GC}^1(v_i), \tag{10}$$

where v is the node features in a graph, and each graph convolution layer GC is defined by:

$$v_i^{l+1} = \text{GC}(v_i^l) = \sum_{j \in \mathcal{N}_v} \mathcal{U}_{\text{mp}}(v_j^l, v_i^l) \mathcal{U}_f(v_j^l), \tag{11}$$

where $\mathcal{N}_v$ is the set of neighbor nodes, $\mathcal{U}_{\text{mp}}$ is the message passing function, and $\mathcal{U}_f$ encodes the neighbor features. Edge-Augmented GNNs (EA-GNNs) Gladstone et al. [2024] improve long-range information flow by introducing edges between distant node pairs. These are often heuristically or randomly added in point clouds, reducing graph diameter. From a deployment perspective, both models are practical for engineering applications, as they can directly leverage existing meshes generated from finite element analysis without requiring additional pre-processing.

4.3 Grid-Based Neural Operators

Fourier Neural Operator (FNO) Li et al. [2020b] initiates the grid-based operator learning approach by performing convolution in the spectral domain, which predicts the PDE solution on a grid by:

$$U = \mathcal{U}_{\text{FL}}^l \circ \mathcal{U}_{\text{FL}}^{l-1} ... \circ \mathcal{U}_{\text{FL}}^1(X), \tag{12}$$

where U is the PDE solution function values evaluated on a set of uniform grid coordinates X, and the Fourier Layer $\mathcal{U}_{\text{FL}}$ is formulated by:

$$G^{l+1} = \mathcal{U}_{\text{FL}}(G^l) = \mathcal{F}^{-1}(R_\phi \cdot \mathcal{F}(G^l)), \tag{13}$$

where G^l is the grid feature output of the l-th Fourier layer, $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier and inverse Fourier transforms, respectively, and R_ϕ is a learned spectral filter. Geometry-Informed Neural Operator (GINO) Li et al. [2023c] extends FNO by integrating point cloud information directly. It uses a Riemann-sum-based discretization to construct the mapping between unstructured point features and grid-aligned features:

$$v_g^1 = \sum_{\|x_g - x_i\|_2 \le r} \mathcal{U}_{\text{mp}}(v_i^0, v_g^0) \mathcal{U}_f(v_i^0), \quad v_g^1 \in G^1, \quad v_g^0 \in G^0, \quad v_i^0 \in X, \tag{14}$$

$$G^l = \mathcal{U}_{\text{FL}}^l \circ \cdots \circ \mathcal{U}_{\text{FL}}^1(G^1), \tag{15}$$

$$v_i^l = \sum_{\|x_g - x_i\|_2 \le r} \mathcal{U}_{\text{mp}}(v_g^l, v_i^0) \mathcal{U}_f(v_g^l), \quad v_i^l \in X^l, \quad v_g^l \in G^l, \tag{16}$$

where r is the hyper-parameter to search the neighbor nodes of each grid point, v_i and v_g represent node and grid features, respectively. However, storing and processing a high-resolution 3D grid is computationally expensive and memory inefficient. To alleviate this, FigConvNet Choy et al. [2025] uses factorized implicit grids instead of explicit 3D grids and processes data with a U-Net backbone, which drastically improves memory efficiency and has demonstrated strong performance on large CFD meshes. From a deployment perspective, although the method requires radius search operations, these can be efficiently executed using hash grid algorithms Sahebi et al. [2025] available in GPU-accelerated libraries such as Warp. Therefore, we consider that both GINO and FigConvNet are suitable for large-scale engineering applications.

4.4 Point-Based Neural Operators

Point-based neural operators aim to directly model solution operators on point clouds. They offer high flexibility and are naturally suited to domains with nonuniform resolution and geometry. PointNet Kashefi and Mukerji [2022] processes unordered sets using shared MLPs and symmetric pooling, which is easily deployed but lacks local structure modeling. More recently, transformer-based architectures Hao et al. [2023] have emerged, predicting PDE solution for one point location with stacked attention layers:

$$u(x_i) = \text{Attn}^l \circ \text{Attn}^{l-1} ... \circ \text{Attn}^1(v_i), \tag{17}$$

where the attention mechanism is formulated as:

$$v_i^{l+1} = \text{Attn}(v_i^l) = \sum_j \frac{\exp(Q_i^l \cdot K_j^l/\tau)}{S_i} \cdot V_j^l, \tag{18}$$

$$S_i = \sum_j \exp(Q_i^l \cdot K_j^l/\tau), \tag{19}$$

where τ is the hyper-parameter, $Q_i^l = \mathcal{U}_q(v_i^l)$, $K_j^l = \mathcal{U}_k(v_j^l)$, and $V_j^l = \mathcal{U}_v(v_j^l)$ represents the queries, keys, and values of the point features. To further enhance the efficiency of transformer nerual operators, Transolver Wu et al. [2024] proposes a linear self-attention mechanism using physics-aware slice tokens. Each point feature v_j is assigned to M slices using a MLP $\mathcal{U}_{\text{slice}}$:

$$\{w_j^i\}_{i=1}^M = \{\text{Softmax}\left(\mathcal{U}_{\text{slice}}^i(v_j)\right)\}_{i=1}^M, \tag{20}$$

$$s^i = \frac{\sum_j w_j^i v_j}{\sum_j w_j^i}, \tag{21}$$

where s^i is the i-th physics token. After the stacked linear attention updates of the physics tokens, the point features is computed by composing the weights and the physics tokens:

$$s^{i,l} = \text{Attn}^l \circ \cdots \circ \text{Attn}^1(s^{i,0}), \tag{22}$$

$$u(x_j) = \sum_i w_j^i \cdot s^{i,l}. \tag{23}$$

Among existing point-based neural operators, we consider that PointNet, GNOT, and Transolver are particularly well-suited for engineering applications, as they can operate directly on raw point cloud data without the need for additional pre-processing.

5 Architectural Enhancement Strategies

Based on the categorization in Section 4, we refer to graph-based, grid-based, and point-cloud-based neural operators as geometric neural operators, as each operates on specific geometric representations. In contrast, branch-trunk architectures are generally designed to model global function spaces, relying heavily on parametric representations of the domain. To ensure a fair and consistent comparison among all selected neural operator models, we need to address several architectural challenges:

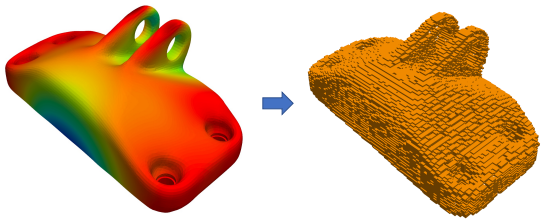
- Many branch-trunk models are limited to problems with explicit parametric representations. Their ability to generalize to freeform geometries remains an open question.
- For problems with parametric design representations, geometric neural operators may be at a disadvantage due to the absence of explicit encoding of these parameters. This makes direct comparison to branch-trunk models potentially unfair.

5.1 Branch-Trunk-Based Models for Freeform Geometries

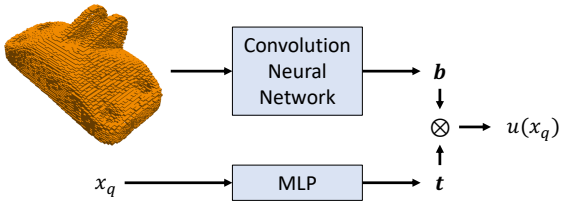
To extend branch-trunk-based models to support freeform geometries, we introduce two strategies for geometry encoding within the branch network:

1. **High-level geometric descriptors:** We compute statistical features from the point cloud representation of the geometry, including centroid, bounding box dimensions, and PCA-derived axes and eigenvalues. These descriptors form a compact feature vector representing the global geometry.

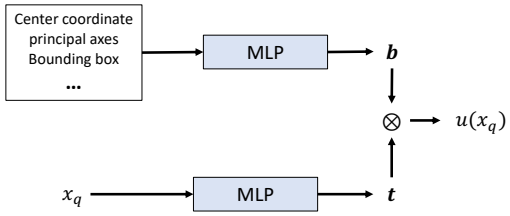
2. **Voxel-based encoding:** The geometry is discretized into a voxel grid, which is processed by a 3D convolutional neural network (CNN) to extract a latent embedding. This representation captures the spatial structure of the geometry in a resolution-aware manner.



(a) Voxel representation of freeform geometry.



(b) DeepONet with voxel-based branch network.



(c) DeepONet with high-level geometric descriptors.

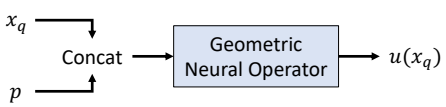
Figure 2: Branch-trunk model architecture enhancements for freeform geometry.

5.2 Parametric Learning in Geometric Neural Operators

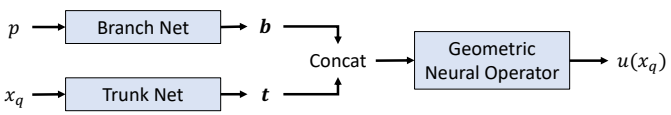
We investigate two approaches to incorporate parametric design representations into geometric neural operators:

1. **Direct concatenation:** The design parameter vector is directly concatenated with the spatial query point x_q and passed as input to the model.

2. **Branch-enhanced integration:** A separate MLP branch network processes the parametric vector, while the trunk network encodes the spatial coordinates. The outputs are fused using concatenation, and the resulting features are passed into the geometric operator.



(a) Direct concatenation.



(b) Branch-enhanced geometric neural operator.

Figure 3: Geometric neural operator architectures incorporating parametric design inputs.

6 Numerical experiment

In this section, we provide a detailed analysis of the benchmarking results across different neural operator families and datasets. Each dataset was split into 60% for training, 10% for validation, and 30% for testing. The models are trained with a learning rate of 0.001, which is adjusted using an adaptive decay schedule to improve convergence stability. Optimization is performed using the Adam optimizer with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. A batch size of 5 is used, and the training process is carried out for 1000 epochs to ensure sufficient convergence. All experiments are conducted on an NVIDIA A100 GPU with 40GB of memory. The model performance is evaluated using the Mean Absolute Error (MAE) of the predicted fluid velocity and pressure. We use the KDTree Ram and Sinha [2019] algorithm to construct graph connectivity for point cloud inputs. For grid-based models, the hidden graph resolution is set to a maximum of 100 voxels to ensure sufficient graph density. We follow the original implementation of each model to set its hyperparameters. In each training step, the gradient is computed based on a single graph. If the hidden dimension exceeds available GPU memory, we reduce it accordingly and always use the largest dimension that fits within the memory constraint.

All models are implemented within a unified experimental framework to ensure fair comparisons across datasets. This framework standardizes architectural depth, hidden dimensions, and spatial resolution. Unless otherwise specified, all models use 3 layers. Hidden dimensions are chosen based on model type: 128 for branch-trunk architectures (e.g., DeepONet, Geom-DeepONet), 32 for graph-based operators (e.g., GNO, GNOT, GUNet), 16 for grid-based models (e.g., FNO, FIGConv), and 128 for point-based networks (e.g., GANO, EAGNO). Each model receives input features defined on voxelized or point-sampled geometries, with spatial resolution set by the dataset-specific grid shape. The input parameter dimensions vary across datasets, spanning 0 to 22 for geometric parameters and 0 to 101 for load parameters. For output activation, we apply the sigmoid function to datasets with normalized outputs (Heat sink, Bracket, Bracket-time) and use the identity function for datasets with unbounded or highly variable outputs (JEB, DrivAer, DrivAer++).

Due to the significant computational cost of training on high-resolution 3D data, it is infeasible to perform exhaustive hyperparameter tuning for each model-dataset pair. To address this, we employ a two-stage configuration strategy. First, we perform a grid search over a subset of each dataset, training and validating on the same split to identify a suitable combination of hidden dimension and layer count. Once the best configuration is selected, it is fixed and applied in all subsequent evaluations for that model. This procedure ensures a balance between computational feasibility and robust, standardized benchmarking across diverse neural operator architectures. The complete spatial configurations of all datasets are summarized in Table 2.

Table 2: Dataset configuration used across all model evaluations.

Dataset	**Geo Dim**	**Load Dim**	x **range**	y **range**	z **range**	**Grid Shape**
Heat sink	4	0	[0.0, 1.0]	[0.0, 1.0]	[0.0, 1.0]	(50, 50, 50)
Bracket	3	1	[0.0, 1.0]	[0.0, 1.0]	[0.0, 1.0]	(50, 50, 50)
Bracket-time	0	101	[-0.2, 1.1]	[-0.1, 0.4]	[0.0, 0.2]	(65, 25, 10)
JEB	0	0	[0.0, 1.0]	[0.0, 1.0]	[0.0, 1.0]	(50, 50, 50)
DrivAer	22	0	[-1.2, 4.1]	[-1.1, 1.1]	[0.0, 1.8]	(53, 22, 18)
DrivAer++	0	0	[-1.2, 4.1]	[-1.1, 1.1]	[0.0, 1.8]	(53, 22, 18)

The results are presented in three steps: (1) evaluation of branch–trunk neural operators with parametric inputs, (2) evaluation of general-geometry neural operators (vanilla versions) without parametric information, and (3) evaluation of enhanced variants of both branch–trunk and general-geometry models.

6.1 Main results

Table 3 compares the performance of branch-trunk neural operators on datasets featuring both explicit parametric geometry representation and free-form geometries. Notably, DCON, with its proposed stacked operator layer, achieves the lowest errors on the Heat sink (0.10%) and Bracket (1.75%) datasets. S-NOT, which incorporates temporal attention encoding in the branch network, delivers the best performance on the time-dependent Bracket-time dataset (5.6%). GANO utilizes parametric representations when available and point cloud representations when parametric inputs are not provided. This explains the significant performance difference observed between the DrivAer and DrivAer++

datasets. We find that for complex vehicle geometries, point cloud representations lead to better performance than parametric representations.

Table 3: Test errors (%) of branch-trunk neural operators. Bold indicates best in each column.

Model	Heat sink	Bracket	Bracket-time	JEB	DrivAer	DrivAer++
DeepONet	0.15%	5.90%	14.8%	-	53.1%	-
Geom-DeepONet	0.18%	1.84%	15.1%	-	51.3%	-
S-DeepONet	-	-	8.6%	-	-	-
S-NOT	-	-	**5.6%**	-	-	-
DCON	**0.10%**	**1.75%**	10.7%	-	47.4%	-
GANO	0.16%	1.86%	12.9%	**47.1%**	**23.1%**	**21.8%**

Table 4 compares the performance of general geometric neural operators on different time-independent datasets **without parametric inputs**, relying solely on mesh or point cloud representations. A key observation is that no single model consistently outperforms others across all datasets, underscoring the diversity of our benchmark and its utility for evaluating generalization in neural operator design. For datasets with parametric inputs and relatively simple geometries (e.g., Heat sink and Bracket), geometric neural operators are generally less competitive than branch-trunk models. However, for datasets with freeform geometries such as JEB and DrivAer++, geometric models perform significantly better.

In comparing different model types, we observe that GNO consistently yields the highest error across all datasets. Additionally, grid-based models outperform point-based models on the Heat sink and JEB datasets but underperform on Bracket. We hypothesize that this is due to the greater variation in geometry size within the Heat sink and JEB datasets compared to Bracket. These results suggest that grid-based models may be more robust when geometry sizes vary significantly. Conversely, for the DrivAer and DrivAer++ datasets, point-based models consistently achieve superior performance, indicating their strength in capturing complex PDE fields on intricate geometries.

Table 4: Test errors (%) of general-geometry models without parametric inputs.

Model	Heat sink	Bracket	JEB	DrivAerNet	DrivAer++
Best branch-trunk	**0.10% (DCON)**	**1.75% (DCON)**	47.1% (GANO)	23.1% (GANO)	21.8% (GANO)
GNO	5.69%	10.14%	61.1%	45.8%	33.8%
EA-GNO	6.12%	10.88%	58.6%	48.7%	35.8%
MechGraphNet	5.94%	11.27%	62.9%	49.6%	46.5%
GI-FNO	1.21%	3.94%	31.2%	25.6%	19.8%
FigConvUNet	0.89%	3.80%	**29.8%**	23.7%	18.3%
PointNet	6.14%	8.75%	40.1%	28.5%	18.8%
GNOT	5.55%	3.58%	37.6%	17.9%	18.3%
Transolver	5.23%	2.74%	36.6%	**16.7%**	**17.3%**

6.2 Performances of enhanced models

Table 5 shows the performance of general-geometry models enhanced through the direct concatenation of parametric inputs. This fusion strategy significantly improves prediction accuracy across most models. For example, **GINO** improves from 3.94% to 0.90% on the Bracket dataset, while **Transolver** achieves a test error of 0.32%, outperforming all other models. **PointNet** and **GNOT** also benefit substantially from the inclusion of parametric design information. However, for problems involving time-dependent components, incorporating a dedicated temporal module in the machine learning model remains essential.

Table 5: Test errors (%) of general-geometry models with parametric input concatenation.

Model	Heat sink	Bracket	Bracket-time
Best branch-trunk	**0.10% (DCON)**	1.75% (DCON)	**5.6% (S-NOT)**
GNO	0.11%	1.20%	14.2%
EA-GNO	0.14%	1.79%	13.8%
MechGraphNet	0.20%	23.7%	16.5%
GI-FNO	0.14%	0.90%	12.9%
FigConvUNet	0.13%	0.81%	11.8%
PointNet	0.12%	0.75%	10.2%
GNOT	0.14%	0.45%	11.8%
Transolver	0.11%	**0.32%**	10.3%

As shown in Table 6, further improvements are achieved by separately processing parametric vectors through a dedicated MLP branch. This branch-enhanced fusion strategy consistently improves performance across all models. **FigConvUNet** and **Transolver** exhibit the most significant gains, reducing their Bracket-time errors to 9.8% and 7.9%, respectively. The results from Table 5 and Table 6 indicate that, for problems with parametric representations, existing geometric neural operators can achieve accuracy levels comparable to branch-trunk models by effectively fusing local coordinate information with global parametric features. However, their performances still cannot overcome the temporal attention mechanism.

Table 6: Test errors (%) with branch-enhanced parametric fusion. Values shown as baseline $\rightarrow$ improved.

Model	Heat sink	Bracket-time
Best branch-trunk	0.10% (DCON)	**5.6% (S-NOT)**
GNO	0.11% $\rightarrow$ 0.09%	14.2% $\rightarrow$ 11.4%
EA-GNO	0.14% $\rightarrow$ 0.12%	13.8% $\rightarrow$ 11.2%
MechGraphNet	0.20% $\rightarrow$ 0.15%	16.5% $\rightarrow$ 13.0%
GI-FNO	0.14% $\rightarrow$ 0.08%	12.9% $\rightarrow$ 10.2%
FigConvUNet	**0.13%** $\rightarrow$ **0.07%**	11.8% $\rightarrow$ 9.8%
PointNet	0.12% $\rightarrow$ 0.10%	10.2% $\rightarrow$ 8.3%
GNOT	0.14% $\rightarrow$ 0.11%	11.8% $\rightarrow$ 8.9%
Transolver	0.11% $\rightarrow$ 0.09%	10.3% $\rightarrow$ 7.9%

Table 7 investigates the effect of different geometry encodings for DeepONet on freeform datasets. High-level feature representations lead to poor accuracy, whereas voxel-based CNN encodings improve performance as voxel resolution increases. However, even with the highest resolution setup (100^3), voxel-based DeepONet remains less accurate than the best-performing geometric neural operators. This highlights a fundamental limitation of branch-trunk models when applied to freeform geometries.

Table 7: Test errors (%) of DeepONet with different geometric encodings on freeform datasets.

Model Variant	JEB	DrivAer++
Best model	**29.8% (FigconvUNet)**	**16.7% (transolver)**
High-level features	56.4%	67.2%
Voxel 10^3	47.4%	46.3%
Voxel 50^3	42.9%	43.9%
Voxel 100^3	38.9%	40.7%

We also compare the memory and computational costs of model training by measuring the training time per epoch for each model across all datasets. Table 8 summarizes these results. Models such as DeepONet, Geom-DeepONet,

and PointNet exhibit significantly faster training times, particularly on smaller datasets like Heat sink and Bracket. In contrast, graph-based models are generally slower, reflecting the increased computational complexity associated with graph construction and neighbor searches. Notably, DCON achieves competitive training speeds without compromising predictive accuracy, making it a balanced choice in terms of both efficiency and performance. In terms of the inference, all the model achieved at least 100x faster than Heat sink dataset and at least 1000x faster than other dataset.

Table 8: Training time per epoch (in seconds) and number of trainable parameters for all models across datasets.

Model	**Heat sink**	**Bracket**	**Bracket-time**	**JEB**	**DrivAer**	**DrivAer++**
DeepONet	3.68s (133.25K)	20.5s (133.25K)	75.0s (145.66K)	-	2160s (135.55K)	-
Geom-DeepONet	3.78s (100.22K)	16.6s (100.35K)	79.4s (125.57K)	-	900s (102.53K)	-
S-DeepONet	-	-	74.3s (427.24K)	-	-	-
DCON	3.94s (100.35K)	21.0s (100.35K)	73.4s (112.77K)	-	2320s (102.66K)	-
GANO	3.79s (215.81K)	19.6s (282.11K)	78.0s (320.64K)	389s (215.68K)	2057s (218.11K)	750s (215.68K)
GNO	45.0s (212.16K)	230s (212.19K)	372s (215.39K)	1200s (212.16K)	3600s (212.16K)	3600s (212.16K)
EA-GNO	52.0s (212.16K)	408s (212.19K)	375s (215.39K)	419s (212.16K)	3800s (212.16K)	3800s (212.16K)
GINO	48.0s (162.15M)	211s (162.15M)	98.4s (26.10M)	194s (76.11M)	4670s (38.16M)	467s (38.16M)
FigconvUNet	42.0s (79.43M)	197s (79.43M)	89.4s (77.75M)	235s (99.86M)	4320s (81.73M)	4120s (81.73M)
PointNet	3.70s (247.94K)	16.9s (248.06K)	83.9s (260.87K)	17.4s (247.94K)	771s (247.94K)	750s (247.94K)
GNOT	43.0s (3.67M)	235s (3.67M)	72.9s (3.88M)	283s (3.67M)	3600s (3.88M)	3420s (3.88M)
Transolver	49.0s (585.30K)	228s (585.56K)	75.3s (611.16K)	297s (585.30K)	3600s (581.10K)	3600s (581.10K)

Combining all the analyses above, we summarize our key findings as follows:

- **For problems with parametric representations**, branch-trunk models should be the first choice due to their low training cost and strong performance on structured geometries. For example, DCON consistently achieves the best results on the Heat sink (0.10%) and Bracket (1.75%) datasets (Table 3), while maintaining fast per-epoch training times (Table 8).
- **Even when parametric inputs are available**, point-based models may be preferable when dealing with complex geometries or solutions with abrupt spatial changes. As shown in Table 3, transolver outperforms all other models on the DrivAer (16.7%) and DrivAer++ (17.3%) datasets, both of which feature freeform vehicle geometries.
- **When geometry size varies significantly across the dataset**, grid-based neural operators provide more robust performance. For instance, FigConvUNet achieves the best results on the Heat sink (0.89%) and JEB (29.8%) datasets (Table 4), where geometry scale variation is pronounced.

- **For time-dependent problems**, Models equipped with explicit temporal encoding mechanisms can achieve strong performance, even when the overall architecture is as simple as a branch-trunk design. This is evident in the superior performance of S-DeepONet (8.6% error) compared to all other models on the Bracket-time dataset (Table 5), owing to its GRU-based temporal encoder. The performance of SNOT further validate our conclusion.

7 Conclusion

In this study, we introduced a comprehensive benchmarking framework for evaluating neural operator architectures on six industry-scale 3D engineering design datasets, covering diverse physical domains such as thermal analysis, elasto-plasticity, time-dependent mechanics, and computational fluid dynamics. Our benchmark encompassed twelve neural operator models spanning four architectural categories: branch-trunk, graph-based, grid-based, and point-based. To ensure a fair comparison across domains and geometries, we implemented enhancements that extend branch-trunk models to freeform geometries and enable geometric neural operators to incorporate parametric inputs.

Our findings demonstrate that no single neural operator architecture consistently outperforms others across all tasks. Instead, model performance is closely tied to the alignment between architectural design, input representation, and the underlying physics of each problem. Branch-trunk models—particularly DCON and S-DeepONet—excel in problems with structured parametric inputs and temporal dynamics. Grid-based models like FigConvNet show strong performance on geometrically complex domains due to their spatial regularity and scalability. Point-based models such as Transolver perform robustly across both structured and unstructured settings, leveraging their physics-aware attention mechanisms for flexible representation learning. Despite the breadth of our study, several limitations remain:

- **Model Coverage:** While we benchmarked twelve models, the landscape of neural operator architectures is evolving rapidly. Future work should explore recently proposed models, particularly those designed for multi-physics coupling, uncertainty-aware learning, and advanced spatiotemporal modeling.
- **Dataset Scope:** Our benchmark includes six datasets covering key engineering domains, but it remains limited in breadth. Extending the framework to include problems such as fluid-structure interaction (FSI), contact mechanics, turbulence modeling, and additive manufacturing would enhance generalizability.
- **Scalability and Deployment:** Although we proposed deployment-oriented modifications, our current evaluation does not fully address scalability in real-world settings. Future studies should assess performance under constraints such as limited GPU memory, multi-node distributed environments, and hardware heterogeneity (e.g., CPU vs GPU inference). Testing on high-performance machines with larger memory would be particularly useful for evaluating deployment viability.
- **Interpretability and Robustness:** The interpretability of neural operator predictions and their robustness to noise, mesh perturbations, or out-of-distribution (OOD) inputs remain open challenges. These aspects are especially important for trustworthy adoption in safety-critical engineering applications.
- **Uncertainty Quantification:** Our current benchmarking framework does not account for uncertainty in model predictions. Incorporating principled uncertainty quantification methods—such as Bayesian neural operators or ensemble-based approaches Zhong and Meidani [2023]—is essential for risk-aware design and decision-making.
- **Data Efficiency and Generalization:** While our benchmark emphasizes accuracy and scalability, further analysis is needed to evaluate data efficiency and model generalization under low-data regimes or with varying boundary conditions. This is particularly relevant for applications with expensive or sparse simulation data.

Acknowledgment

This work used the *Delta* advanced computing resources provided by the University of Illinois Urbana-Champaign (UIUC) through the National Center for Supercomputing Applications (NCSA). We gratefully acknowledge the support of these resources.

References

Benedikt Alkin, Andreas Fürst, Simon Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37:25152–25194, 2024.

Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024.

Chris Choy, Alexey Kamenev, Jean Kossaifi, Max Rietmann, Jan Kautz, and Kamyar Azizzadenesheli. Factorized implicit global convolution for automotive computational fluid dynamics prediction. *arXiv preprint arXiv:2502.04317*, 2025.

Ştefan Cobzaş, Radu Miculescu, Adriana Nicolae, et al. *Lipschitz functions*. Springer, 2019.

W. Diab and M. Al Kobaisi. U-deeponet: U-net enhanced deep operator network for geologic carbon sequestration. *Scientific Reports*, 2024.

Moshe Eliasof, Eldad Haber, and Eran Treister. Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations. In *NeurIPS*, 2021. URL `https://proceedings.neurips.cc/paper/2021/file/1f9f9d8ff75205aa73ec83e543d8b571-Paper.pdf`.

Mohamed Elrefaie, Angela Dai, and Faez Ahmed. Drivaernet: A parametric car dataset for data-driven aerodynamic design and graph-based drag prediction. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 88360, page V03AT03A019. American Society of Mechanical Engineers, 2024a.

Mohamed Elrefaie, Florin Morar, Angela Dai, and Faez Ahmed. Drivaernet++: A large-scale multimodal car dataset with computational fluid dynamics simulations and deep learning benchmarks. *Advances in Neural Information Processing Systems*, 37:499–536, 2024b.

Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. *Handbook of numerical analysis*, 7: 713–1018, 2000.

Kailun Feng, Weizhuo Lu, and Yaowu Wang. Assessing environmental performance in early building design stage: An integrated parametric design and machine learning method. *Sustainable Cities and Society*, 50:101596, 2019.

Rini Jasmine Gladstone, Helia Rahmani, Vishvas Suryakumar, Hadi Meidani, Marta D'Elia, and Ahmad Zareei. Mesh-based gnn surrogates for time-independent pdes. *Scientific reports*, 14(1):3394, 2024.

S. Goswami, K. Kontolati, M. D. Shields, and G. E. Karniadakis. Deep transfer operator learning for partial differential equations. *Nature Machine Intelligence*, 2022.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pages 12556–12569. PMLR, 2023.

Junyan He, Seid Koric, Diab Abueidda, Ali Najafi, and Iwona Jasiuk. Geom-deeponet: A point-cloud-based deep operator network for field predictions on 3d parameterized geometries. *Computer Methods in Applied Mechanics and Engineering*, 429:117130, 2024a.

Junyan He, Shashank Kushwaha, Jaewan Park, Seid Koric, Diab Abueidda, and Iwona Jasiuk. Sequential deep operator networks (s-deeponet) for predicting full-field solutions under time-dependent loads. *Engineering Applications of Artificial Intelligence*, 127:107258, 2024b.

Seongjun Hong, Yongmin Kwon, Dongju Shin, Jangseop Park, and Namwoo Kang. Deepjeb: 3d deep learning-based synthetic jet engine bracket dataset. *Journal of Mechanical Design*, 147(4), 2025.

Vishal Jagota, Aman Preet Singh Sethi, and Khushmeet Kumar. Finite element method: an overview. *Walailak Journal of Science and Technology (WJST)*, 10(1):1–8, 2013.

Ali Kashefi and Tapan Mukerji. Physics-informed pointnet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries. *Journal of Computational Physics*, 468:111510, 2022.

Jean Kossaifi, Nikola Kovachki, Kamyar Azizzadenesheli, and Anima Anandkumar. Multi-grid tensorized fourier neural operator for high-resolution pdes. *arXiv preprint arXiv:2310.00120*, 2023.

Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22(290):1–76, 2021.

S. Lanthaler, R. Molinaro, P. Hadorn, and S. Mishra. Nonlinear reconstruction for operator learning of pdes with discontinuities. *arXiv preprint arXiv:2210.01074*, 2022.

Seungjun Lee and Taeil Oh. Inducing point operator transformer: A flexible and scalable architecture for solving pdes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 153–161, 2024.

Kangjie Li and Wenjing Ye. D-fno: A decomposed fourier neural operator for large-scale parametric partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 436:117732, 2025.

Z. et al. Li. Porous-deeponet: Learning the solution operators of parametric pdes in porous media. *Advances in Water Resources*, 2024.

Zijie Li, Dule Shu, and Amir Barati Farimani. Scalable transformer for pde surrogate modeling. *Advances in Neural Information Processing Systems*, 36:28010–28039, 2023a.

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

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020b.

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

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Animashree Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *NeurIPS*, 2020d. URL `https://proceedings.neurips.cc/paper/2020/file/4b21cf96d4cf612f239a6c322b10c8fe-Paper.pdf`.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Karthik Bhattacharya, Andrew Stuart, and Animashree Anandkumar. Fourier neural operator: A general architecture for parametric partial differential equations. *Journal of Computational Physics*, 2021.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023b.

Zongyi Li, Nikola Kovachki, Chris Choy, et al. Geometry-informed neural operator for large-scale 3d pdes. In *NeurIPS*, 2023c. URL `https://proceedings.neurips.cc/paper/2023/file/70518ea42831f02afc3a2828993935ad-Paper-Conference.pdf`.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3):1–27, 2024.

Ning Liu, Siavash Jafarzadeh, and Yue Yu. Domain agnostic fourier neural operators. *Advances in Neural Information Processing Systems*, 36:47438–47450, 2023a.

Ning Liu, Siavash Jafarzadeh, and Yue Yu. Domain agnostic fourier neural operators. *Advances in Neural Information Processing Systems*, 36:47438–47450, 2023b.

Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. *arXiv preprint arXiv:2402.16845*, 2024.

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations. *arXiv preprint arXiv:1910.03193*, 2019.

L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 2021a.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3): 218–229, March 2021b. ISSN 2522-5839. doi:10.1038/s42256-021-00302-5. URL http://dx.doi.org/10.1038/s42256-021-00302-5.

Lu Lu, Xuhui Meng, Shengze Cai, Zhiping Mao, Somdatta Goswami, Zhongqiang Zhang, and George Em Karniadakis. A comprehensive and fair comparison of two neural operators (with practical extensions) based on fair data. *Computer Methods in Applied Mechanics and Engineering*, 393:114778, 2022a.

Yijun Lu, Wei Wu, Xuechuan Geng, Yanchen Liu, Hao Zheng, and Miaomiao Hou. Multi-objective optimization of building environmental performance: An integrated parametric design method based on machine learning approaches. *Energies*, 15(19):7031, 2022b.

L. Mandl, S. Goswami, L. Lambers, and T. Ricken. Separable deeponet: Breaking the curse of dimensionality in physics-informed machine learning. *arXiv preprint arXiv:2407.15887*, 2024.

Oisín M Morrison, Federico Pichi, and Jan S Hesthaven. Gfn: A graph feedforward network for resolution-invariant reduced operator learning in multifidelity applications. *Computer Methods in Applied Mechanics and Engineering*, 432:117458, 2024.

Stanley Osher and Ronald P Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational physics*, 169(2):463–502, 2001.

J. Park and N. Kang. Point-deeponet: A deep operator network integrating pointnet for nonlinear analysis of non-parametric 3d geometries and load conditions. *SSRN Electronic Journal*, 2025.

Albert E Patterson and James T Allison. Mapping and enforcement of minimally restrictive manufacturability constraints in mechanical design. *ASME Open Journal of Engineering*, 1, 2022.

Kai Qi and Jian Sun. Gabor-filtered fourier neural operator for solving partial differential equations. *Computers & Fluids*, 274:106239, 2024.

Yuan Qiu, Nolan Bridges, and Peng Chen. Derivative-enhanced deep operator network. *Advances in Neural Information Processing Systems*, 37:20945–20981, 2024.

Parikshit Ram and Kaushik Sinha. Revisiting kd-tree for nearest neighbor search. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining*, pages 1378–1388, 2019.

Amin Sahebi, Marco Procaccini, and Roberto Giorgi. Hashgrid: an optimized architecture for accelerating graph computing on fpgas. *Future Generation Computer Systems*, 162:107497, 2025.

Louis Serrano, Patrik Damberg, Yizhi Ouyang, Andrea Coviello, Rui Chao, and Liangliang Sun. Operator learning with neural fields: Tackling pdes on general geometries. In *NeurIPS*, 2023a. URL https://proceedings.neurips.cc/paper/2023/hash/df54302388bbc145aacaa1a54a4a5933-Abstract-Conference.html.

Louis Serrano, Lise Le Boudec, Armand Kassaï Koupaï, Thomas X Wang, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Operator learning with neural fields: Tackling pdes on general geometries. *Advances in Neural Information Processing Systems*, 36:70581–70611, 2023b.

Louis Serrano, Lise Le Boudec, Armand Kassaï Koupaï, Thomas X Wang, Yuan Yin, Jean-Noël Vittaut, and Patrick Gallinari. Operator learning with neural fields: Tackling pdes on general geometries. *Advances in Neural Information Processing Systems*, 36:70581–70611, 2023c.

Louis Serrano, Thomas X Wang, Etienne Le Naour, Jean-Noël Vittaut, and Patrick Gallinari. Aroma: Preserving spatial structure for latent pde modeling with local neural fields. *Advances in Neural Information Processing Systems*, 37: 13489–13521, 2024.

Louis et al. Serrano. Representation equivalent neural operators: a framework for alias-free operator learning. In *NeurIPS*, 2023. URL https://papers.nips.cc/paper_files/paper/2023/file/dc35c593e61f6df62db541b976d09dcf-Paper-Conference.pdf.

Mauricio Velasco, Bernardo Rychtenberg, and Soledad Villar. Graph neural networks and non-commuting operators. In *NeurIPS*, 2024. URL https://proceedings.neurips.cc/paper/2024/hash/ad9964b731bc7b5621a83c7869fc653b-Abstract-Conference.html.

Chuang et al. Wang. Latent neural operator for solving forward and inverse pdes. *Advances in Neural Information Processing Systems*, 2024a. URL https://proceedings.neurips.cc/paper/2024/hash/39f6d5c2e310a5a629dcfc4d517aa0d1-Abstract-Conference.html.

H. et al. Wang. α-vi deeponet: A prior-robust variational bayesian approach for operator learning. *arXiv preprint arXiv:2408.00681*, 2024b.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Yaseen Wu et al. Vcc: Scaling transformers to 128k tokens or more by prioritizing important tokens. In *NeurIPS*, 2023. URL https://proceedings.neurips.cc/paper/2023/hash/4054556fcaa934b0bf76da52cf4f92cb-Abstract-Conference.html.

V. et al. Yadav. Digital twins for nuclear safeguards and security: Assessment of challenges, opportunities, and current state-of-practice. *U.S. Nuclear Regulatory Commission Report*, 2023.

Y. Yang. Deeponet for solving nonlinear partial differential equations with physics-informed training. *arXiv preprint arXiv:2410.04344*, 2024.

Yue Yu, Ning Liu, Fei Lu, Tian Gao, Siavash Jafarzadeh, and Stewart A Silling. Nonlocal attention operator: Materializing hidden knowledge towards interpretable physics discovery. *Advances in Neural Information Processing Systems*, 37:113797–113822, 2024a.

Yue Yu, Ning Liu, Fei Lu, Tian Gao, Siavash Jafarzadeh, and Stewart A Silling. Nonlocal attention operator: Materializing hidden knowledge towards interpretable physics discovery. *Advances in Neural Information Processing Systems*, 37:113797–113822, 2024b.

J. et al. Zhang. Derivative-enhanced deep operator network. *NeurIPS 2024 Conference Paper*, 2024.

Z. Zhang, C. Moya, L. Lu, G. Lin, and H. Schaeffer. Deeponet as a multi-operator extrapolation model: Distributed pretraining with physics-informed fine-tuning. *SSRN Electronic Journal*, 2025.

Jianwei Zheng, Wei Li, Ni Xu, and Junwei Zhu. Alias-free mamba neural operator. In *NeurIPS*, 2024a. URL https://proceedings.neurips.cc/paper/2024/file/5ee553ec47c31e46a1209bb858b30aa5-Paper-Conference.pdf.

Jianwei Zheng, Wei Li, Ni Xu, Junwei Zhu, and Xiaoqin Zhang. Alias-free mamba neural operator. *Advances in Neural Information Processing Systems*, 37:52962–52995, 2024b.

Weiheng Zhong and Hadi Meidani. Pi-vae: Physics-informed variational auto-encoder for stochastic differential equations. *Computer Methods in Applied Mechanics and Engineering*, 403:115664, 2023.

Weiheng Zhong and Hadi Meidani. Physics-informed discretization-independent deep compositional operator network. *Computer Methods in Applied Mechanics and Engineering*, 431:117274, 2024.

Weiheng Zhong and Hadi Meidani. Physics-informed geometry-aware neural operator. *Computer Methods in Applied Mechanics and Engineering*, 434:117540, 2025.