arXiv:2411.15178v1 [cs.LG] 18 Nov 2024

Harnessing Scale and Physics: A Multi-Graph Neural Operator Framework for PDEs on Arbitrary Geometries

Zhihao Li
zli416@connect.hkust-gz.edu.cn
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China

Haoze Song
hsong492@connect.hkust-gz.edu.cn
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China

Di Xiao
shawd@buaa.edu.cn
Beihang University
Beijing, China

Zhilu Lai
zhilulai@ust.hk
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China

Wei Wang*
weiwcs@ust.hk
The Hong Kong University of Science and Technology (Guangzhou)
Guangzhou, China

ABSTRACT

Partial Differential Equations (PDEs) underpin many scientific phenomena, yet traditional computational approaches often struggle with complex, nonlinear systems and irregular geometries. This paper introduces the **AMG** method, a **M**ulti-**G**raph neural operator approach designed for efficiently solving PDEs on **A**rbitrary geometries. AMG leverages advanced graph-based techniques and dynamic attention mechanisms within a novel GraphFormer architecture, enabling precise management of diverse spatial domains and complex data interdependencies. By constructing multi-scale graphs to handle variable feature frequencies and a physics graph to encapsulate inherent physical properties, AMG significantly outperforms previous methods, which are typically limited to uniform grids. We present a comprehensive evaluation of AMG across six benchmarks, demonstrating its consistent superiority over existing state-of-the-art models. Our findings highlight the transformative potential of tailored graph neural operators in surmounting the challenges faced by conventional PDE solvers. Our code and datasets are available on https://github.com/lizhihao2022/AMG.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**.

KEYWORDS

Partial differential equations, neural operator, geometric leaning

ACM Reference Format:
Zhihao Li, Haoze Song, Di Xiao, Zhilu Lai, and Wei Wang. 2025. Harnessing Scale and Physics: A Multi-Graph Neural Operator Framework for PDEs on Arbitrary Geometries. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Conference acronym 'XX, June 03–05, 2018, Woodstock, NY
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Partial Differential Equations (PDEs) underlie critical phenomena across diverse fields, from fluid dynamics to quantum mechanics, showcasing their ability to model complex variable relationships. Traditional approaches, while foundational, often struggle with the complexities and non-linearities of these systems, particularly in irregular geometries. Operator Learning has emerged as a transformative approach, leveraging deep learning architectures like Deep Operator Networks (DeepONet) and Fourier Neural Operators (FNO) to directly map input conditions to PDE solutions [16, 18, 25]. This method stands out for its versatility, requiring no retraining for different conditions, thus enabling efficient model adaptation across diverse settings.

However, applying these deep learning techniques to real-world problems poses significant challenges, especially with irregular geometries where traditional methods like FNO [18], using Fast Fourier Transform, and U-Net [32], employing convolutions, are inherently limited to uniform grids. Innovations such as Graph Neural Operators (GNO) [22], and geometric adaptations like Geo-FNO [17], attempt to address these limitations by projecting irregular domains into more manageable latent meshes. Moreover, GINO combines these approaches to enhance modeling across various scales [23]. However, the inherent limitations of Fourier bases, particularly under the periodic boundary assumption, lead to significant performance degradation in complex geometries [9]. Similarly, graph kernels often fail to capture global information effectively [13, 37, 38].

The challenge extends to the domain's frequency spectrum; high-frequency areas require more intensive learning efforts compared to the predominantly flat, low-frequency regions. This disparity in learning demand across different frequencies, akin to techniques used in computer vision for tasks such as super-resolution, underscores the potential of tailoring neural network structures to leverage these differences for enhanced performance [10, 34].

In PDE learning, multi-scale methods have shown promise; however, they are generally limited to uniform grids and often require

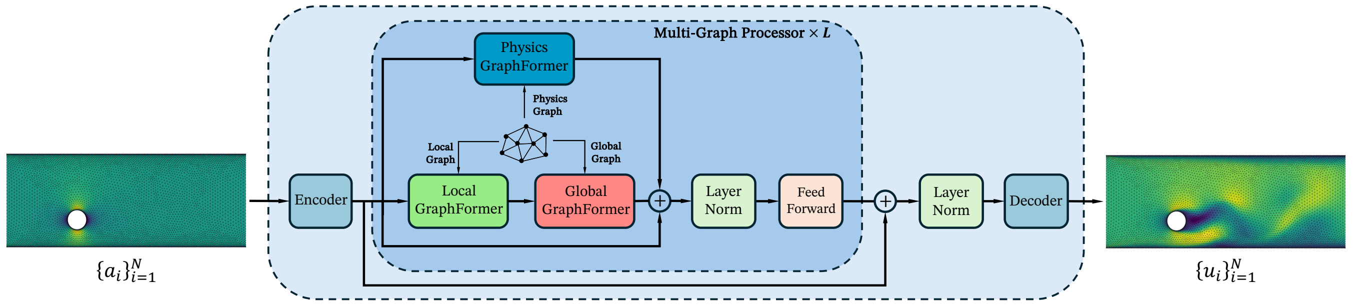


Figure 1: Overview of the model architecture.

dimensions to be powers of two, particularly in methods utilizing wavelets [11, 19, 20]. Furthermore, transformer-based models, which distribute sample points across equal-sized attention grids, struggle with the highly variable frequency behaviors seen in phenomena like wave propagation [4, 13, 37, 38]. Existing graph-based methods, which often employ a k-Nearest-Neighbor approach to construct graphs, treat all nodes equally, neglecting the diverse learning needs inherent in PDE dynamics. This one-size-fits-all approach to node degrees can significantly impede the efficacy of graph-based models.

To address the challenges associated with solving PDEs on arbitrary geometries and capturing multi-scale features, we introduce **AMG**—a method employing a **M**ulti-scale **G**raph neural operator tailored for **A**rbitrary geometries. AMG leverages three distinct types of graphs, constructed from the given coordinates: two are designed for multi-scale processing is dedicated to capturing the underlying physical properties. At the heart of AMG is the GraphFormer, a novel architecture equipped with dynamic graph attention mechanisms. This design allows the model to compute the hidden representations of each node by dynamically attending to its neighbors, effectively handling the complex interdependencies within the data. Furthermore, we theoretically establish that graph attention can be seen as a learnable integral, enhancing our method's ability to model continuous spaces. Extensive experiments were conducted on six benchmarks, including four well-established ones and two custom-designed to test our method's efficacy across diverse geometries and dynamic mesh configurations. AMG consistently outperforms existing solutions, achieving a remarkable relative gain on six benchmarks.

The contributions of this paper are summarized as follows:

- We use local sampling (Section 3.2.3) and global sampling (Section 3.2.4) to construct multi-scale graphs for capturing different frequencies of features and a physical graph (Section 3.2.5) to encode inherent physical properties effectively.
- We introduce a GraphFormer (Section 3.4) architecture with dynamic graph attention mechanisms, providing a scalable and flexible encoder-processor-decoder framework (Section 3.1) for learning operators tailored to arbitrary geometries.
- AMG demonstrates superior performance, achieving consistent state-of-the-art results with significant relative gains across a variety of benchmarks.

2 PRELIMINARIES

2.1 Problem Formulation

We consider Partial Differential Equations (PDEs) defined over a domain $D \subset \mathbb{R}^d$. Define $\mathcal{A} = \mathcal{A}(D;\mathbb{R}^{d_a})$ and $\mathcal{U} = \mathcal{U}(D;\mathbb{R}^{d_u})$ as two Sobolev spaces $\mathcal{H}^{s,p}$, with parameters $s > 0$ and $p \geq 1$. Our aim is to learn an operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$, mapping from the input function space $\mathcal{A}$ to the solution function space $\mathcal{U}$. Specifically, we select $s > 0$ and $p = 2$ to take advantage of the Hilbert space structure, which facilitates the definition of projections.

The operator $\mathcal{G}$ is characterized as an integral operator with a kernel κ, where $\kappa : D \times D \to L^2$, and is formalized by the integral equation:

$$\mathcal{G}a(\mathbf{x}) = \int_D \kappa(\mathbf{x}, \mathbf{y})a(\mathbf{y})\, d\mathbf{y}, \tag{1}$$

enabling $\mathcal{G}$ to operate within the Hilbert space framework and leveraging the properties of L^2 spaces for enhanced analytical and computational efficiency.

2.2 Graph Neural Operators

A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ includes nodes $\mathcal{V} = \{1, ..., n\}$ and edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, with $(j, i) \in \mathcal{E}$ representing an edge from node j to node i. Each node $i \in \mathcal{V}$ initially possesses a representation $\mathbf{h}_i^{(0)} \in \mathbb{R}^{d_h}$. An undirected graph is depicted using bidirectional edges between nodes.

A Graph Neural Network (GNN) layer updates the representation of each node by aggregating information from its neighbors. The input to a GNN layer consists of a set of node representations $\{\mathbf{h}_i \in \mathbb{R}^{d_h} \mid i \in \mathcal{V}\}$ and the set of edges $\mathcal{E}$. The output is a new set of node representations $\{\mathbf{h}'_i \in \mathbb{R}^{d_h} \mid i \in \mathcal{V}\}$, where each node's updated state is calculated as:

$$\mathbf{h}'_i = f_\theta(\mathbf{h}_i, \text{AGGREGATE}(\{\mathbf{h}_j \mid j \in \mathcal{N}_i\})) \tag{2}$$

The function f and the aggregation method AGGREGATE largely define the distinctions among various GNN architectures.

If we accurately construct the graph on the spatial domain D of the PDE, the kernel integration can be interpreted as an aggregation of messages [22]. In the edge-conditioned aggregation mechanism [8, 35] utilized at the k^{th} layer, the output feature $\mathbf{h}_i^k$ for node i is calculated based on its previous feature $\mathbf{h}^{k-1}$ as follows:

$$u(\mathbf{x}_i) = \mathbf{h}_i^k = \frac{\sum_{j \in \mathcal{N}(i)} \exp\left(\kappa(\mathbf{x}_j, \mathbf{x}_i)\right)\mathbf{h}_j^{k-1}}{\sum_{j \in \mathcal{N}(i)} \exp\left(\kappa(\mathbf{x}_j, \mathbf{x}_i)\right)}, \tag{3}$$

where $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a parameterized kernel function that evaluates the correlation between node pairs (j, i), influencing the strength of the message passed from node j to node i.

3 METHODOLOGY

This section outlines the AMG method's architecture for solving PDEs on arbitrary geometries. We start with an architecture overview in Section 3.1, followed by the construction of multi-graphs in Section 3.2. The processing module is detailed in Section 3.3, and the GraphFormer Block is discussed in Section 3.4.

3.1 Overview of Model Architecture

Our model architecture is designed to effectively handle the complexities involved in processing partial differential equations (PDEs) on irregular geometries using graph neural networks. The architecture is illustrated in Figure 1.

Input and Encoder: The process begins with the input dataset $\{a_i\}_{i=1}^N$, which consists of the initial conditions or parameters for the PDEs. These inputs are first processed by an Encoder, which transforms the raw data into a preliminary feature representation suitable for further processing within the neural network.

Graph Construction: Two distinct types of graphs, the Physics Graph and the Multi-Scale Graph, are constructed from the encoded features. The Physics Graph encapsulates the underlying physical laws governing the phenomena being modeled, while the Multi-Scale Graph captures interactions at various scales, crucial for accurately modeling complex systems.

Multi-Graph Processing Layers: The core of the architecture comprises multiple processing layers, each consisting of a GraphFormer, Message Passing, and Layer Normalization components followed by a Feed Forward network. The GraphFormer component applies transformations specific to graph data, facilitating the propagation and update of node features. Message Passing enables the exchange of information between nodes, enhancing the model's ability to learn from the topology of the graph. Layer Normalization is employed to stabilize the learning process, and the Feed Forward networks provide additional transformation capabilities to the node features.

Decoder and Output: Following the processing layers, a Decoder reverts the graph-based features back into the spatial domain, generating the output $\{u_i\}_{i=1}^N$ which represents the solution to the PDE at the discretized points. This output effectively demonstrates the model's capability to predict complex phenomena governed by the PDEs.

This architecture leverages the strengths of graph neural networks to process data over irregular domains, ensuring robustness and accuracy in capturing the dynamics of the system modeled.

3.2 Graph Construction

3.2.1 High-Frequency Indicator. To efficiently and effectively pinpoint regions within feature maps that contain detail-rich information, we introduce a high-frequency indicator. This indicator is designed to rapidly identify high-frequency areas that are crucial for accurate PDE solutions in complex geometries. Given a feature map of point set $F \in \mathbb{R}^{N \times C}$,where C is the number of feature channels and a specific down-sampling ratio s, the high frequency indicator per node, $H_F \in \mathbb{R}^N$, is computed as follows:

$$H_F = \sum_{c=1}^{C} \left| F^{(c)} - (F^{(c)})_{\downarrow s \uparrow s} \right|, \tag{4}$$

where $(F^{(c)})_{\downarrow s \uparrow s}$ denotes the channel c of the feature map F after it has been bilinearly down-sampled and subsequently up-sampled by a factor of s. The choice of s effectively balances detail preservation with minimal information loss, optimizing the process for quick and effective high-frequency detection.

3.2.2 Partitioning of Point Sets. To efficiently manage the challenge of generating overlapping partitions within a point set, we define each partition as a neighborhood ball within Euclidean space, characterized by centroid location and scale. We utilize a farthest point sampling (FPS) algorithm [31] to select centroids in a manner that ensures even coverage across the entire point set. This method starts with an arbitrary initial point and iteratively selects subsequent points that are the farthest in metric distance from all previously selected points. This approach not only guarantees a superior coverage compared to random sampling but also takes into account the spatial distribution of the points, contrasting with methods like convolutional neural networks (CNNs), that process data agnostically of spatial distributions. Consequently, our graph construction strategy establishes receptive fields in a data-dependent manner, significantly enhancing the model's capability to capture and process complex spatial relationships. Detailed description of the FPS algorithm is provided in Appendix A.1.

3.2.3 Local Sampling. Local sampling is essential for constructing graphs that precisely capture the immediate neighborhood dynamics of a node, particularly valuable in systems characterized by complex local interactions, such as in solving PDEs with intricate local behaviors. This method selects each node v and its neighbors $\mathcal{N}(v)$ based on a high-frequency indicator (Eq.(4)), identifying nodes that reside within areas of high detail and information density.

In practice, nodes are chosen for their rich detail using the high-frequency indicator to determine the composition of each local graph. For every node v, its neighbors $\mathcal{N}(v)$ are selected to ensure that only nodes within areas of substantial detail and information content are included. This targeted selection strategy guarantees that the connections within the graph are meaningful and representative of significant local interactions. By focusing on nodes with high information content, the model can adapt more effectively to variations in data density and local structural complexities. This approach is invaluable in scenarios demanding high precision in local detail, enabling the model to accurately capture complex dynamics.

To maintain local coherence, we use the Euclidean distance between two nodes as the metric for establishing connections:

$$d_{\text{local}}(i, j) = \|\mathbf{p}_i - \mathbf{p}_j\|_2, \tag{5}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the position vector of nodes i and j, respectively. This metric also allows for connecting nodes that belong to different domains.

An illustrative example of this local sampling process and the resultant graph structure is shown in Figure 2(b). This visualization underscores how local sampling concentrates on areas rich in detail, thereby substantially enhancing the model's capacity to learn nuanced behaviors and interactions, which are crucial for solving complex PDEs. Detailed description of the local sampling algorithm is provided in Appendix A.2.

3.2.4 Global Sampling. Conversely, global sampling aims to capture the broader structure of the dataset by establishing connections across wider spatial extents. This sampling strategy selects nodes that span the entire domain in a pattern designed to maximize coverage and minimize redundancy, typically implemented using a dilated sampling technique as introduced in Section 3.2.2. By increasing the intervals between selected nodes, this method provides a comprehensive overview of global interactions essential for understanding large-scale trends and phenomena.

Connectivity among nodes is established through a metric that measures the similarity in node features, employing cosine similarity:

$$d_{\text{global}}(i,j) = \frac{\mathbf{h}_i \cdot \mathbf{h}_j}{\|\mathbf{h}_i\|\|\mathbf{h}_j\|}, \tag{6}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ are the feature vectors of nodes i and j, respectively. This metric ensures that nodes with similar features are linked, regardless of their physical placement, thus enhancing the graph's ability to model phenomena accurately.

An illustrative example of this global sampling process is shown in Figure 2(b). The combination of both local and global sampling strategies in graph construction allows the model to effectively balance detail-oriented and holistic learning objectives.

3.2.5 Physics Graph. In our model, node attributes are embedded into higher dimensions, allowing us to interpret them as high-level physical attributes, which are often the solutions sought in complex physical systems. These high-level attributes derive from more fundamental, lower-level physical properties.

To effectively represent these relationships, we construct a physics graph where each node corresponds to a fundamental physical attribute. In this graph, nodes from the standard operational graph are linked to all nodes in the physics graph, emphasizing the foundational contributions of these attributes to higher-level phenomena. The edges within the physics graph symbolize the interactions between each of these lower-level physical attributes.

In scenarios where explicit physical information about the connections between attributes is available, the physics graph is constructed based on this empirical data. However, in cases where such specifics are lacking, we opt for a fully-connected graph configuration. This approach is justified by the typically small number of lower-level nodes and the frequent interactions among them, ensuring comprehensive coverage of potential influences and interactions within the system.

An illustration of the construction of the physics graph is shown in Figure 2(a), and the detailed propagation mechanisms between the physics graph and the multi-scale graph are defined in Section 3.3.2.

3.3 Multi-Graph Processor

Multi-Scale Graph Blocks integrate both local and global-scale information crucial for effective operator learning. These blocks compute local and global graphs per block based on the inputs, allowing for dynamic graph updates throughout the model's processing.

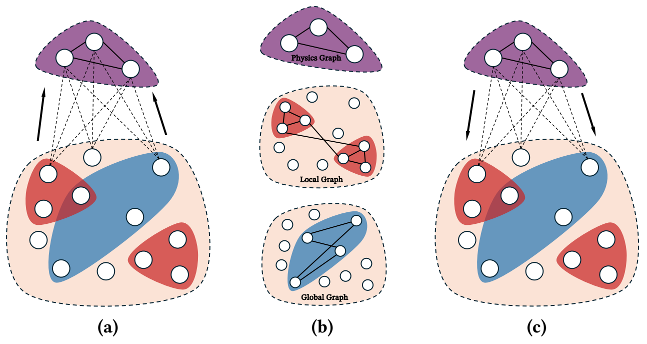


Figure 2: (a) Message passing from multi-scale graph to physics graph. (b) Three types of graphs as the input of GraphFormer. (c) Message passing from physics graph to multi-scale graph.

The design of the Multi-Scale Graph Blocks follows the general MetaFormer structure [42], employing GraphFormers to perform graph aggregation acting as token mixers. Depending on the type of graph received, a GraphFormer can process either local or global information, enhancing its flexibility and capability to handle complex data structures.

3.3.1 Cross-Scale Graph Aggregation. After constructing two graphs at different scales, as shown in Figure 2(b), nodes are processed through the GraphFormer Block, defining local or global edges accordingly:

$$\{\mathbf{h}_i^{local}\}_{i=1}^N = \text{GraphFormer}(\{\mathbf{h}_i^x\}_{i=1}^N, \mathcal{E}_{local}), \tag{7}$$

$$\{\mathbf{h}_i^{global}\}_{i=1}^N = \text{GraphFormer}(\{\mathbf{h}_i^{local}\}_{i=1}^N, \mathcal{E}_{global}), \tag{8}$$

where $\mathbf{h}_i^x$ represents the input features for node i, and $\mathcal{E}_{local}$ and $\mathcal{E}_{global}$ are the sets of edges in the local and global graphs, respectively.

3.3.2 Physics Graph Propagation. As depicted in Figure 2(a), virtual physics nodes are initialized by aggregating information from the original graph nodes. Specifically, the value of virtual physics nodes $\{\mathbf{h}_j^v\}_{j=1}^M$ is calculated as:

$$\mathbf{h}_j^v = \sum_{i=1}^N \mathbf{e}_{ij}^v \mathbf{h}_i^x, \tag{9}$$

where $\mathbf{h}_i^x$ are the nodes from the original graph, and $\mathbf{e}_{ij}^v$ are the edge weights defined by:

$$\mathbf{e}_{ij}^v = \frac{\mathbf{W}_v \mathbf{h}_i^x}{\sum_{i' \in \mathcal{N}_j^v} \mathbf{h}_{i'}^x}, \tag{10}$$

with $\mathbf{W}_v \in \mathbb{R}^{d_h \times d_h}$ being a trainable linear layer. Subsequently, the transformed physical nodes are processed through the GraphFormer:

$$\{\mathbf{h'}_j^v\}_{j=1}^M = \text{GraphFormer}(\{\mathbf{h}_j^v\}_{j=1}^M, \mathcal{E}_{phy}), \tag{11}$$

Finally, these physical nodes are reintegrated into the graph nodes through a message-passing scheme, effectively blending the computed physical node representations back into the global node features:

$$\mathbf{h'}_i^x = \mathbf{h}_i^{global} + \sum_{j=1}^{M} \mathbf{e}_{ij}^v \mathbf{h'}_j^v, \tag{12}$$

where each token $\mathbf{h'}_j^v$ is broadcasted to all graph nodes during the calculation. This method ensures that both local and global information is incorporated into each node, facilitating a comprehensive learning of the PDE operators across different scales and physical interpretations.

3.4 GraphFormer Block

Inspired by the general MetaFormer architecture [42], we introduce the GraphFormer block, which replaces traditional attention mechanisms with graph attention, thereby tailoring the approach to handle graph-based data more effectively. As depicted in Figure 3, this block employs graph aggregation techniques as token mixers, facilitating efficient information processing across the network.

3.4.1 Graph Attention Mechanism. A scoring function $f : \mathbb{R}_h^d \times \mathbb{R}_h^d \to \mathbb{R}$ evaluates the importance of features from neighbor j to node i in every edge (j, i):

$$f(\mathbf{h}_i, \mathbf{h}_j) = \text{LeakyReLU}\left(\mathbf{a}^\top \cdot \left[\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j\right]\right), \tag{13}$$

where $\mathbf{a} \in \mathbb{R}^{2d_h}$ and $\mathbf{W} \in \mathbb{R}^{d_h \times d_h}$ are trainable parameters, and $\|$ denotes vector concatenation. The attention scores are then normalized using a softmax function across all neighbors $j \in \mathcal{N}_i$:

$$\alpha_{ij} = \text{softmax}_j(f(\mathbf{h}_i, \mathbf{h}_j)) = \frac{\exp(f(\mathbf{h}_i, \mathbf{h}_j))}{\sum_{j' \in \mathcal{N}_i} \exp(f(\mathbf{h}_i, \mathbf{h}_{j'}))}. \tag{14}$$

The Graph Attention (GA) then compute the new representation for node i as a weighted average of transformed features of its neighbors, applying a nonlinearity σ:

$$\mathbf{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \cdot \mathbf{W}\mathbf{h}_j\right). \tag{15}$$

Following [2], we apply the $\mathbf{a}$ layer after the LeakyReLU nonlinearity and $\mathbf{W}$ after concatenation, essentially using an MLP to compute the score for each query-key pair.

The overall complexity is $O(|\mathcal{V}|d_h^2 + |\mathcal{E}|d_h)$, where $|\mathcal{V}|$ represents the number of nodes and $|\mathcal{E}|$ the number of edges in the graphs used, either multi-scale or physics. Since $|\mathcal{V}| \ll N$ and $|\mathcal{E}| = k|\mathcal{V}| \leq |\mathcal{V}|^2$, the computational complexity remains linear with respect to the number of mesh points N. This ensures that the proposed method scales efficiently even as the mesh complexity increases.

3.4.2 Properties of GraphFormer. Prior methods have approached PDE learning as an iterative update process, demonstrating that canonical attention mechanisms can approximate integral operators over the input domain Ω [4, 16]. To deepen our theoretical understanding of the GraphFormer, we propose that it similarly acts as a learnable integral on Ω:

Theorem 1 (GraphFormer as a Learnable Integral on Ω). *Given an input function $a : \Omega \to \mathbb{R}^d$ and a mesh point $\mathbf{x} \in \Omega$, GraphFormer seeks to approximate the integral operator $\mathcal{G}$, defined by:*

$$\mathcal{G}a(\mathbf{x}) = \int_\Omega \kappa(\mathbf{x}, \xi) a(\xi)\, d\xi, \tag{16}$$

where $\kappa(\cdot, \cdot)$ denotes a kernel function over $\Omega \times \Omega$.

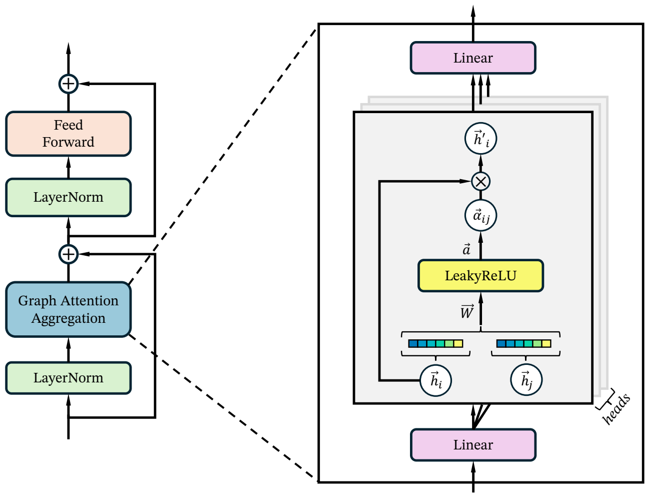


Figure 3: Left: The Architecture of GraphFormer Block. Right: An illustration of multi-head attention by node i on its neighborhood j.

A complete proof is available in Appendix B. This theoretical foundation substantiates the capability of GraphFormer to efficiently learn complex mappings essential for solving PDEs across irregular domains, underscoring its utility in advanced computational mathematics.

4 EXPERIMENTS

4.1 General Setting

In practice, to learn a neural operator, we use a dataset $\mathcal{D} = \{(a, u)\}$, where $u = \mathcal{G}(a)$. Due to challenges in directly representing functions, we discretize the input and solution functions over the domain D using an irregular mesh, as per a specified mesh generation algorithm [27]. Thus, we consider a set of function pairs $(a_i, u_i)_{i=1}^N$, where $a_i = a(\mathbf{x}_i)$ and $u_i = u(\mathbf{x}_i)$ at discretized points $\mathbf{x}_i$ within the domain $D \subset \mathbb{R}^d$.

Our goal is to approximate $\mathcal{G}$ by optimizing the network parameters θ, through the following optimization problem:

$$\min_{\theta \in \Theta} \mathcal{L}(\theta) := \min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \left[\|\tilde{\mathcal{G}}_\theta(a_i) - u_i\|^2\right]. \tag{17}$$

Table 1: Summary of Experiment Benchmarks.

Mesh Type	Benchmarks	Geometry	# Dim	# Mesh	# TimeStep
Fixed	Navier-Stokes	Structured	2	4096	10
	Shape-Net Car	Unstructured	3	32186	/
	Poisson	Unstructured	2	3242	/
	Airfoil	Unstructured	2	5233	10
	Weather Forecast	Unstructured	2	11088	30
Dynamic	Cylinder Flow	Unstructured	2	7209 ~ 7581	100
	Deforming Plate	Unstructured	3	672 ~ 2189	50

Table 2: Main results on benchmarks. A smaller value indicates better performance. For clarity, the best result is in bold and the second best is underlined.

Model	NS	Shape-Net Car		Poisson	Airfoil				DP	Cylinder Flow		
		pressure	velocity		density	pressure	velocity x	velocity y		pressure	velocity x	velocity y
MLP	0.1607	0.2422	0.2428	0.4804	0.0963	0.0947	0.0438	0.0559	0.0856	0.0781	0.0447	0.1753
U-Net	0.0970	0.1380	0.1867	0.2699	0.0579	0.0575	0.0536	0.0523	0.0873	0.0558	0.0224	0.0562
MGN	0.1614	0.2432	0.2477	0.4428	0.0506	0.0491	0.0208	0.0258	0.0323	0.0774	0.0456	0.1715
GNO	0.1610	0.2453	0.2596	0.5167	0.0786	0.0769	0.0286	0.0406	0.0414	0.0855	0.0509	0.1835
ONO	0.0824	0.1115	0.1389	0.0199	0.0097	0.0096	0.0075	0.0088	0.0358	0.0201	0.0157	0.0263
GNOT	0.5805	0.1109	0.1206	0.4403	0.0054	0.0049	0.0044	0.0040	0.2362	0.0392	0.0272	0.0545
LSM	0.0957	0.1169	0.1487	0.2612	0.0305	0.0299	0.0525	0.0341	0.0802	0.0093	0.0065	0.0126
Transolver	0.0593	0.0993	0.1208	0.0162	0.0036	0.0032	0.0028	0.0035	0.0264	0.0092	0.0059	0.0117
AMG(Ours)	**0.0476**	**0.0878**	**0.0919**	**0.0152**	**0.0021**	**0.0020**	**0.0014**	**0.0018**	**0.0257**	**0.0050**	**0.0038**	**0.0078**
Promotion	19.70%	11.64%	23.81%	6.44%	39.97%	38.01%	51.44%	47.45%	2.35%	45.33%	35.44%	32.94%

4.1.1 Benchmarks. We conducted comprehensive analyses across a variety of benchmark scenarios to demonstrate the superiority of our method, as detailed in [17, 18, 29]. These benchmarks were carefully selected to encompass a wide range of PDE problems, from fluid dynamics to structural deformations, using both static and dynamic conditions. Specifically, we developed datasets for the Poisson equation and Cylinder Flow, testing our model on both fixed and dynamic unstructured meshes, as well as standard structured meshes. We also expand our experimental scope to include global weather forecasting, utilizing data from the European Centre for Medium-Range Weather Forecasts (ECMWF). This new experiment involves climate data from the first quarter of 2018, focusing specifically on the atmospheric conditions at a pressure level of 50hPa. This breadth of testing highlights our model's adaptability and precision across various computational environments. For ease of reference, these benchmarks are summarized in Table 1, with visual examples provided and detailed configurations discussed in Appendix C.

4.1.2 Baselines. We conducted a comprehensive evaluation against a variety of established methods to validate our approach. Traditional models like MLP [33] and U-Net [32] were included for their foundational roles in computational tasks. We also assessed against specialized graph-based methods such as MeshGraphNets (MGN) [29] and Graph Neural Operator (GNO) [22], which are designed for complex spatial data. Additionally, our model was compared with advanced transformer-based methods, including LSM [37], GNOT [13], ONO [39] and Transolver [38], to benchmark against the latest approaches in handling high-dimensional data complexities.

4.1.3 Implementation. To ensure fairness in performance comparison, all models were uniformly trained across 500 epochs using the L_2 loss function [18]. We utilized the Adam [15] and AdamW [24] optimizers, initiating with a learning rate of 10^{-3} and implementing a decay factor of $\gamma = 0.5$ to halve the rate every 100 epochs. For handling dynamic mesh sizes, which transformer-based models inherently struggle with, we adopted a padding strategy similar to that used in GNOT [13] to accommodate these models. All experiments were conducted using a single Nvidia A800 80GB GPU, and unless specified, default hyperparameters were employed for the baseline models. This setup facilitated a controlled environment to accurately assess the comparative effectiveness of each method.

4.2 Main Results

AMG's robustness and accuracy across diverse PDE benchmarks are highlighted in the performance comparison (Table 2), demonstrating significant enhancements in both structured and unstructured mesh environments.

4.2.1 Structured Mesh Performance. In the Navier-Stokes benchmark with structured 2D geometry, AMG achieved a notable reduction in errors by 19.70%. This illustrates AMG's effectiveness in structured environments where regular geometries predominate, requiring high precision to accurately model dynamic interactions.

4.2.2 Fixed Unstructured Mesh Analysis. In the Poisson benchmark, AMG improved prediction by 6.44%, demonstrating its effectiveness in modeling static spatial variations in simpler systems. In the Shape-Net Car benchmark involving complex 3D automobile geometries, AMG achieved improvements of 11.64% in pressure accuracy of surface and 23.81% in velocity accuracy of surrounding air, showcasing its capability to handle intricate 3D shapes and multiple attributes in a large unstructured mesh environment. For the Airfoil benchmark, AMG excelled by enhancing velocity predictions by 51.44% for velocity X and 47.45% for velocity Y, along with increases of 39.97% in density and 38.01% in pressure accuracy. This demonstrates its proficiency in dynamic simulations, capturing detailed airflow dynamics and interactions. Overall, these results affirm AMG's robust performance across diverse fixed unstructured mesh scenarios, illustrating its versatility and high precision in handling complex physical phenomena across different dimensions and conditions.

4.2.3 Dynamic Unstructured Mesh Challenges. AMG's prowess was distinctly displayed in the Cylinder Flow and Deforming Plate benchmarks, both demanding high adaptability in dynamic environments. In the 2D Cylinder Flow scenario, AMG significantly enhanced performance metrics, improving pressure accuracy by 45.33%, velocity X by 35.44%, and velocity Y by 32.94% over 100 timesteps. These enhancements confirm AMG's exceptional ability to handle temporal changes and adaptive mesh akin to real-world fluid dynamics. The 3D Deforming Plate benchmark posed additional challenges with both spatial and temporal dynamics. Here, AMG achieved a 2.35% improvement in pressure accuracy, demonstrating its competency in managing the intricacies of 3D dynamic systems that necessitate synchronized predictions across varied scales. Collectively, these benchmarks validate AMG's effectiveness in dynamic unstructured meshes, underlining its high precision and versatility in complex, real-world simulation scenarios. This robust performance underscores AMG's sophisticated capacity to navigate intricate interactions and continual changes.

4.2.4 Three-Dimensional Benchmark Insights. The Shape-Net Car benchmark involves predicting two attributes across a large mesh size of 32,186 points spread over three dimensions. The Deforming Plate benchmark presents a dynamic three-dimensional challenge, where the mesh evolves over time, adding another layer of complexity. In addressing these challenges, AMG leverages a combination of a graph specifically designed to handle arbitrary geometries, and multi-scale and physics graphs that effectively manage features across different scales. This integrated approach significantly enhances AMG's performance, leading to a substantial reduction in prediction errors by 35.44%. This improvement not only highlights the model's scalability but also its adeptness in processing volumetric data, effectively handling the intricacies of varied spatial and temporal scales, thereby markedly reducing prediction errors.

4.2.5 Comparison with Graph-Based Methods. Our analysis reveals that graph-based models such as MeshGraphNets (MGN) and Graph Neural Operator (GNO) generally underperform in static benchmarks like Shape-Net Car and Poisson, where the scenarios are dominated by low-frequency features. These models are better suited to dynamic benchmarks such as Deforming Plate, which involve varying shapes; however, their performance still falls short of expectations. To overcome these limitations, AMG incorporates global and physics graphs. The global graph helps in capturing broader spatial relationships that are crucial for static environments, while the physics graph integrates fundamental physical laws and principles, enhancing the model's ability to accurately predict behaviors in dynamic scenarios. This strategic use of multiple graph types enables AMG to significantly outperform traditional graph-based methods across a range of benchmarks, effectively addressing both static and dynamic challenges.

4.2.6 Comparison with Transformer-Based Methods. Transformer-based models such as Transolver and GNOT previously set the standard for state-of-the-art performance in handling complex data structures. Despite their strengths, AMG has demonstrated superior performance by integrating specialized graph techniques that enhance feature processing capabilities across various scales. Utilizing the graph neural operator's ability to effectively manage local high-frequency features, combined with the global graph and physics graph's adeptness at capturing global low-frequency features, AMG has significantly outperformed these transformer-based models. This remarkable improvement showcases AMG's enhanced capability to address a broader range of dynamic and static phenomena more accurately than the existing state-of-the-art models.

Table 3: Results on Weather Forecasting.

Model	L_2 Error		
	Temperature	Wind U	Wind V
MLP	0.0194	0.3198	0.3585
U-Net	0.0113	0.2279	0.3108
MGN	0.0186	0.3533	0.3559
GNO	0.1173	0.3405	0.3896
ONO	0.0156	0.2535	0.2914
GNOT	0.0364	0.7954	1.0206
LSM	0.0164	<u>0.2095</u>	<u>0.2435</u>
Transolver	<u>0.0077</u>	0.2155	0.2762
AMG (Ours)	**0.0067**	**0.2020**	**0.2335**
Promotion	**↓13.08%**	**↓3.62%**	**↓4.11%**

4.2.7 Experiment on Real-World Dataset. Our objective was to predict the next hour's temperature and wind components (U-component and V-component). The data was sampled from a reduced grid of the original, capturing 11,088 gridpoints, to balance computational demand with meaningful analysis. As illustrated in Table 3, AMG outperforms other methods across all metrics, particularly excelling in dynamic conditions, such as predicting wind directions and speeds. This success underscores AMG's integration of multiscale graphs and dynamic graph attention mechanisms, enhancing its predictive accuracy in complex, variable scenarios.

These results affirm the robustness of AMG across diverse computational settings, from static and dynamic meshes to two-dimensional and three-dimensional spaces. The model's adeptness at navigating various mesh types and geometries, combined with its capability to manage intricate local and global interactions, establishes AMG as a powerful solution for solving PDEs in a wide range of applications. This comprehensive performance, particularly evident in real-world dynamic scenarios like weather forecasting, demonstrates AMG's potential to deliver precise, reliable predictions across different scales and conditions.

Table 4: Ablation Study on Graphs.

Configuration	L_2 Error		
	pressure	velocity x	velocity y
w/o Local Graph	0.0101	0.0076	0.0136
w/o Global Graph	0.0135	0.0084	0.0149
w/o MultiScale Graph	0.0087	0.0070	0.0137
w/o Physics Graph	0.0598	0.0238	0.0622
Baseline (AMG)	**0.0050**	**0.0038**	**0.0078**

4.3 Model Analysis

4.3.1 Ablation Studies on Graph Configurations. We conducted ablation studies to determine the contribution of different graph configurations—Local, Global, Multiscale, and Physics—to our model's performance. These studies, summarized in Table 4, show the impact of each graph type on the accuracy of predictions for pressure, velocity X, and velocity Y.

- **Without Local Graph:** Removing this graph increased L2 errors, especially in velocity Y, underscoring its role in capturing local interactions.
- **Without Global Graph:** The absence led to higher L2 errors, highlighting its importance for contextual relationship understanding.
- **Without Multiscale Graph:** Its removal caused minor performance drops, indicating its role in linking local and global insights.
- **Without Physics Graph:** This had the most significant impact, emphasizing its necessity for incorporating physical principles.

The baseline AMG configuration outperformed all variations, illustrating the critical need for an integrated approach to handle complex simulations effectively.

4.3.2 Hyperparameter Analysis. Our comprehensive hyperparameter study, summarized in Table 5, identifies optimal configurations that significantly enhance model performance across various metrics. The analysis revealed that simply increasing the number of layers, node numbers, or head numbers does not consistently reduce L_2 errors, indicating that optimal hyperparameter settings are crucial for maximizing performance. These results demonstrate that effective hyperparameter tuning is crucial for deploying AMG in diverse computational environments. By strategically selecting settings tailored to specific tasks, we can refine the model's architecture, thereby enhancing both performance and operational efficiency. The insights gained from this study are instrumental in fine-tuning the model to achieve superior performance across different applications.

Table 5: Hyperparameter Study on Cylinder Flow.

Type	Configuration	L_2 Error		
		Pressure	Velocity X	Velocity Y
Layer Number	layer=1	0.0148	0.0104	0.0188
	layer=2	0.0106	0.0073	0.0151
	layer=3	**0.0050**	**0.0038**	**0.0078**
	layer=4	0.0096	0.0068	0.0141
Local Node Number	n=256	0.0096	0.0068	0.0141
	n=512	0.0107	0.0082	0.0148
	n=1024	**0.0050**	**0.0038**	**0.0078**
	n=2048	0.0114	0.0077	0.0149
Global Sample Ratio	r=12.5%	0.0166	0.0109	0.0178
	r=25%	**0.0050**	**0.0038**	**0.0078**
	r=50%	0.0091	0.0069	0.0128
Physical Node Number	p=8	0.0171	0.0121	0.0230
	p=16	0.0142	0.0104	0.0204
	p=32	**0.0050**	**0.0038**	**0.0078**
	p=64	0.0159	0.0116	0.0222
Head Number	h=1	0.0139	0.0092	0.0185
	h=4	0.0096	0.0066	0.0121
	h=8	**0.0050**	**0.0038**	**0.0078**
	h=12	0.0096	0.0065	0.0122
Global Graph Degree	global k=2	0.0131	0.0095	0.0193
	global k=4	**0.0050**	**0.0038**	**0.0078**
	global k=6	0.0090	0.0065	0.0115
	global k=8	0.0179	0.0113	0.0233
Local Graph Degree	local k=2	0.0152	0.0103	0.0176
	local k=4	0.0138	0.0111	0.0218
	local k=6	**0.0050**	**0.0038**	**0.0078**
	local k=8	0.0101	0.0071	0.0150

4.3.3 Multi-Scale Sampling. Multi-Scale Sampling is a critical technique in our methodology, enabling the AMG model to effectively capture and integrate diverse spatial information. This approach involves two distinct sampling strategies: global sampling and local sampling. Global sampling (Figure 4(a,c) focuses on capturing broad-scale data integration across the entire domain, while local sampling (Figure 4(b,d)) targets high-resolution detail capture within specific regions of interest. These techniques collectively enhance the model's ability to process and analyze data from complex PDE systems with varying dynamics and geometric irregularities.

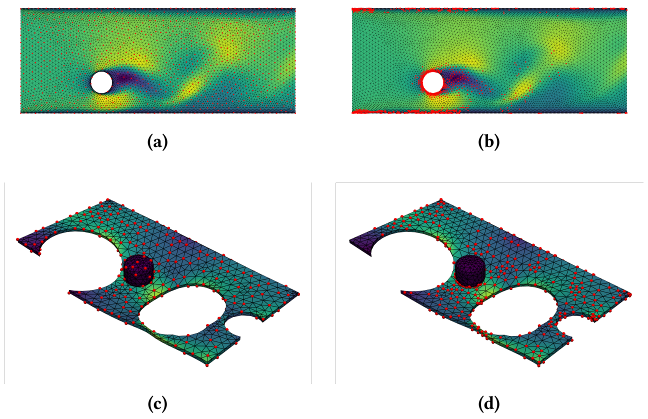


Figure 4: Illustration of global sampling and local sampling in Cylinder Flow (a, b) and Deforming Plate (c, d).

5 RELATED WORK

5.1 Neural Operators

Neural Operators utilize deep neural networks to efficiently solve complex PDE systems by learning mappings between two functional spaces [16]. Typically modeled as kernel integral operators, various parameterization strategies have been developed, including the widely recognized Fourier Neural Operators (FNO) which integrate Fast Fourier Transform (FFT) [18].

5.1.1 Graph-based Neural Operators. Graph-based neural operators such as GNO [22] utilize Message Passing Graph Networks to process data in irregular domains but face challenges like prolonged training and difficulty in capturing global low-frequency features [19, 23, 36, 37, 41]. MGNO [19] attempts to overcome these by using multi-scale graphs, yet its reliance on pre-defined matrix decomposition restricts flexibility with dynamic meshes and complex geometries. Recent models like GINO and GeoFNO [17, 23] combine graph-based strategies with Fourier Neural Operators to adapt irregular data into uniform grids, aiming to resolve some of the enduring challenges of GNO. However, these advancements still contend with foundational issues inherent in the original graph-based methods.

5.1.2 Transformer-based Neural Operators. Transformer-based neural operators utilize attention mechanisms to encode operators in latent spaces effectively. The Galerkin Transformer [4] uses a Galerkin-type linear attention for operator parameterization, while ONO [39] introduces an orthogonal attention module inspired by

Mercer's theorem to refine the kernel integral. OFormer [21] employs a standard transformer structure with cross-attention, encoding both input functions and query locations into a latent ODE representation. GNOT [13] and the state-of-the-art Transolver [38] further enhance these concepts. GNOT employs heterogeneous normalized cross-attention for uniform feature handling, and Transolver leverages physics-aware tokens for more precise operator learning. Despite their advantages, transformer models can struggle with detailed point-level attention, which may result in missing high-frequency information in complex PDE environments.

5.1.3 Multi-level Neural Operators. Multi-level neural operators are adept at representing PDE behavior across varying scales, enhancing performance significantly. The Multiwavelet Transform (MWT) [11] uses hierarchical multiwavelet projections to handle kernel operator singularities and signal fluctuations. Advanced multigrid techniques like MgNO [14] and M2NO [20] incorporate the V-cycle of traditional multigrid methods into neural operator learning, with M2NO also integrating features from algebraic multigrid and multiresolution wavelet transform for superior performance. While these methods excel in managing resolution scales, they face challenges in irregular domains and can be computationally demanding. This underscores the need for further refinement to ensure both efficiency and broad applicability in complex PDE scenarios.

5.2 Geometric Deep Learning

Geometric Deep Learning is pivotal for managing irregular geometries, leveraging advances outlined in [3]. Graph Neural Networks (GNNs) are central to this domain, utilizing kernels on connected graphs for effective representation learning, as demonstrated in [7, 12, 29]. Additionally, PointNet [30, 31] and Point Transformer [6] effectively handle scattered point clouds.

For PDE learning, the MAgNet architecture [1] merges coordinate-based methods with GNNs, enhancing performance in irregular meshes. Implicit neural representations for PDE dynamics forecasting, notably [40] and [5], enhance the modeling capabilities in dynamic systems. The work of [28] provides a mesh-agnostic approach to dimensionality reduction, particularly beneficial for handling spatio-temporal data in continuous domains. The Graph Neural Operator (GINO) [23] and Geo-FNO [17] adapt irregular grids into more uniform latent structures, optimizing the application of Fourier neural operators. However, transformer-based models like LSM [37], GNOT [13], and Transolver [38] still face significant challenges with arbitrary geometries and high-frequency detail capture, which are crucial for accurately solving PDEs across various scales.

6 CONCLUSION AND FUTURE WORK

This paper introduces AMG, a novel Multi-Graph neural operator method designed to efficiently solve Partial Differential Equations (PDEs) on Arbitrary geometries. AMG harnesses advanced graph-based techniques combined with dynamic graph attention mechanisms within an innovative GraphFormer architecture. This approach enables the precise handling of diverse spatial domains and intricate data interdependencies, setting it apart from previous methods constrained to uniform grids.

Our extensive evaluations across various benchmarks confirm AMG's superior performance and scalability compared to existing state-of-the-art solutions. These results demonstrate AMG's potential to significantly advance the computational mathematics field, especially in applications involving complex and non-linear systems.

Looking ahead, we aim to explore the possibilities of large-scale pre-training of AMG, similar to the development of foundation models in other domains of AI. Such advancements could establish a new paradigm for PDE solving, potentially leading to more generalized and robust methods capable of tackling a broader range of scientific and engineering challenges. This direction not only promises to enhance the model's performance but also explores its adaptability and applicability to increasingly complex scenarios.

REFERENCES

[1] Oussama Boussif, Dan Assouline, Loubna Benabbou, and Yoshua Bengio. 2022. MAgNet: Mesh Agnostic Neural PDE Solver. *ArXiv* abs/2210.05495 (2022).

[2] Shaked Brody, Uri Alon, and Eran Yahav. 2022. How Attentive are Graph Attention Networks?. In *ICLR*. OpenReview.net.

[3] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2016. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Process. Mag.* 34 (2016), 18–42.

[4] Shuhao Cao. 2021. Choose a Transformer: Fourier or Galerkin. In *Neural Information Processing Systems.*

[5] Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, G. A. Pershing, Henrique Teles Maia, Maurizio M. Chiaramonte, Kevin T. Carlberg, and Eitan Grinspun. 2023. CROM: Continuous Reduced-Order Modeling of PDEs Using Implicit Neural Representations. In *ICLR*. OpenReview.net.

[6] Nico Engel, Vasileios Belagiannis, and Klaus C. J. Dietmayer. 2020. Point Transformer. *IEEE Access* 9 (2020), 134826–134840.

[7] Hongyang Gao and Shuiwang Ji. 2019. Graph U-Nets. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44 (2019), 4948–4960.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 1263–1272.

[9] David Gottlieb and Steven A. Orszag. 1977. *Numerical Analysis of Spectral Methods.* Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611970425 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611970425

[10] Yuanbiao Gou, Peng Hu, Jiancheng Lv, Hongyuan Zhu, and Xiaocui Peng. 2023. Rethinking Image Super Resolution from Long-Tailed Distribution Learning Perspective. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023), 14327–14336.

[11] Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. 2021. Multiwavelet-based Operator Learning for Differential Equations. In *NeurIPS*. 24048–24062.

[12] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Neural Information Processing Systems.*

[13] Zhongkai Hao, Chengyang Ying, Zhengyi Wang, Hang Su, Yinpeng Dong, Songming Liu, Ze Cheng, Jun Zhu, and Jian Song. 2023. GNOT: A General Neural Operator Transformer for Operator Learning. *ArXiv* abs/2302.14376 (2023).

[14] Juncai He, Xinliang Liu, and Jinchao Xu. 2024. MgNO: Efficient parameterization of linear operators via multigrid. *ArXiv* abs/2310.19809 (2024).

[15] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR (Poster)*.

[16] Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. 2023. Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs. *J. Mach. Learn. Res.* 24 (2023), 89:1–89:97.

[17] Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. 2024. Fourier neural operator with learned deformations for PDEs on general geometries. *J. Mach. Learn. Res.* 24, 1, Article 388 (mar 2024), 26 pages.

[18] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. 2021. Fourier Neural Operator for Parametric Partial Differential Equations. In *ICLR*. OpenReview.net.

[19] Zongyi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew M. Stuart, Kaushik Bhattacharya, and Anima Anandkumar. 2020. Multipole Graph Neural Operator for Parametric Partial Differential Equations. In *NeurIPS.*

[20] Zhihao Li, Zhilu Lai, Xiaobo Wang, and Wei Wang. 2024. M2NO: Multiresolution Operator Learning with Multiwavelet-based Algebraic Multigrid Method. *ArXiv* abs/2406.04822 (2024). https://api.semanticscholar.org/CorpusID:270357841

[21] Zijie Li, Kazem Meidani, and Amir Barati Farimani. 2023. Transformer for Partial Differential Equations' Operator Learning. *Transactions on Machine Learning Research* (2023).

[22] Zong-Yi Li, Nikola B. Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew M. Stuart, and Anima Anandkumar. 2020. Neural Operator: Graph Kernel Network for Partial Differential Equations. *ArXiv* abs/2003.03485 (2020).

[23] Zong-Yi Li, Nikola B. 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. *ArXiv* abs/2309.00583 (2023). https://api.semanticscholar.org/CorpusID:261494027

[24] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *ICLR (Poster)*. OpenReview.net.

[25] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. 2021. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nat. Mach. Intell.* 3, 3 (2021), 218–229.

[26] COMSOL Multiphysics. 1998. Introduction to COMSOL multiphysics®. *COMSOL Multiphysics, Burlington, MA, accessed Feb* 9 (1998), 2018.

[27] Steven J. Owen. 1998. A Survey of Unstructured Mesh Generation Technology. In *International Meshing Roundtable Conference*. https://api.semanticscholar.org/CorpusID:2675840

[28] Shaowu Pan, Steven L. Brunton, and J. Nathan Kutz. 2023. Neural Implicit Flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data. *J. Mach. Learn. Res.* 24 (2023), 41:1–41:60.

[29] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. 2021. Learning Mesh-Based Simulation with Graph Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=roNqYL0_XP

[30] C. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2016. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 77–85.

[31] C. Qi, L. Yi, Hao Su, and Leonidas J. Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:1745976

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *MICCAI (3) (Lecture Notes in Computer Science, Vol. 9351)*. Springer, 234–241.

[33] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323 (1986), 533–536.

[34] Jin shan Pan, Sifei Liu, Deqing Sun, Jiawei Zhang, Yang Liu, Jimmy S. J. Ren, Zechao Li, Jinhui Tang, Huchuan Lu, Yu-Wing Tai, and Ming-Hsuan Yang. 2018. Learning Dual Convolutional Neural Networks for Low-Level Vision. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), 3070–3079.

[35] Martin Simonovsky and Nikos Komodakis. 2017. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. In *CVPR*. IEEE Computer Society, 29–38.

[36] Tapas Tripura and Souvik Chakraborty. 2023. Wavelet Neural Operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering* 404 (2023), 115783.

[37] Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. 2023. Solving High-Dimensional PDEs with Latent Spectral Models. *ArXiv* abs/2301.12664 (2023).

[38] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. 2024. Transolver: A Fast Transformer Solver for PDEs on General Geometries. *ArXiv* abs/2402.02366 (2024).

[39] Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. 2024. Improved Operator Learning by Orthogonal Attention. *ArXiv* abs/2310.12487 (2024).

[40] Yuan Yin, Matthieu Kirchmeyer, Jean-Yves Franceschi, Alain Rakotomamonjy, and Patrick Gallinari. 2023. Continuous PDE Dynamics Forecasting with Implicit Neural Representations. In *ICLR*. OpenReview.net.

[41] Huaiqian You, Yue Yu, Marta D'Elia, Tian Gao, and Stewart Silling. 2022. Nonlocal kernel network (NKN): A stable and resolution-independent deep neural network. *J. Comput. Phys.* 469 (2022), 111536.

[42] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. 2022. MetaFormer is Actually What You Need for Vision. In *CVPR*. IEEE, 10809–10819.

A SAMPLING ALGORITHM DETAILS

A.1 Farthest Point Sampling Algorithm

Farthest Point Sampling (FPS) [6] is a technique used to select a subset of points from a larger set in such a way that the selected points are as far apart from each other as possible. This method is particularly useful in scenarios where uniform coverage across the entire dataset is crucial. As detailed in Algorithm 1, given a set of input points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, FPS iteratively selects points $\{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_m}\}$, ensuring that each new point $\mathbf{x}_{i_j}$ is the farthest away from previously selected points $\{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_{j-1}}\}$. This strategy enhances coverage and is particularly advantageous over random sampling, especially in applications where data distribution is non-uniform.

Unlike convolutional neural networks (CNNs), which process data without considering underlying spatial distributions, FPS takes a data-dependent approach to create receptive fields, thus preserving important spatial hierarchies within the data. This method is highly effective for constructing graphs or other data structures where the geometric arrangement of the points is significant.

Algorithm 1 Farthest Point Sampling Algorithm

```
1:  Input: Coordinates Set $C$ of shape $(N, D)$, where $N$ is the number of nodes and $D$ is the dimensionality and $n$, the number of nodes to sample.
2:  Initialize: $minDistances[N] \leftarrow \{\infty\}$;
3:  Initialize: $sampledIndices[n] \leftarrow \{-1\}$;
4:  $current \leftarrow \text{randint}(0, N-1)$;    ▷ Randomly select the first
5:  $sampledIndices[0] \leftarrow current$;
6:  for $i \leftarrow 1$ to $n-1$ do
7:      for $j \leftarrow 0$ to $N-1$ do
8:          $distance \leftarrow \text{computeDistance}\ (C[j], C[current])$;
9:          if $distance < minDistances[j]$ then
10:             $minDistances[j] \leftarrow distance$;
11:         end if
12:     end for
13:     $current \leftarrow \text{argmax}(minDistances)$;
14:     $SampledIndices[i] \leftarrow current$;
15: end for
16: Output: $sampledIndices$
```

A.2 Local Sampling Algorithm

Local sampling plays a pivotal role in constructing graphs that capture the immediate neighborhood dynamics around nodes, crucial for addressing complex local interactions in PDE-solving environments. This method uses a high-frequency indicator (refer to Eq.(4)) to selectively sample nodes that are rich in detail, thereby ensuring the inclusion of only the most informative nodes in each local graph.

The sampling procedure involves computing a feature map that highlights areas of high detail. Nodes are then selected based on their prominence in this feature map, and the local connectivity among these nodes is established via a nearest neighbor approach using their positions. This ensures that each node's local environment is effectively represented, capturing significant local interactions that are essential for precise PDE solutions.

The Local Sampling Algorithm, outlined in Algorithm 2, describes the process of selecting nodes and constructing a local graph. The algorithm takes as input the node features and their coordinates, processes them in batches, and performs local sampling and graph construction for each batch independently.

The topK function selects the n highest values from the feature map within each batch, ensuring that nodes with the highest detail are chosen. The K-nearest neighbors (knn) function establishes edges based on the proximity of these selected nodes, effectively capturing local structural information. This targeted sampling and graph construction are vital for adapting the learning process to complex data variations and local interactions, thereby enhancing the model's ability to solve PDEs with high precision.

Algorithm 2 Local Sampling Algorithm

```
1:  Input:
2:    $X$: Node Set of shape $(N, D_x)$, where $N$ is the number of nodes and $D_x$ is the dimensionality of node features.
3:    $C$: Coordinates Set of shape $(N, D_{pos})$, where $D_{pos}$ is the dimensionality of positions.
4:    $batch$: Array of batch indices corresponding to each node.
5:    $n$: Number of nodes to sample per batch.
6:    $k$: Number of nearest neighbors to consider.
7:  Initialize: $sampledIndices \leftarrow []$;
8:  Initialize: $sampledEdgeIndex \leftarrow []$;
9:  $featureMap \leftarrow$ computeFeatureMap$(X, C, batch)$;
10: for $i \leftarrow 1$ to $batch$ do
11:     $mask \leftarrow (batch == i)$;
12:     $sampledIndices[i] \leftarrow$ topK$(FeatureMap[mask], n)$;
13:     $sampledEdgeIndex[i] \leftarrow$ knn$(C[mask], C[mask], k)$;
14: end for
15: Output: $sampledIndices$, $sampledEdgeIndex$
```

B PROOF OF THEOREM 1

The theorem is established by demonstrating that the graph attention mechanism employed in the GraphFormer can be formalized as a Monte-Carlo approximation of an integral operator [4, 16, 38].

Proof. Given an input function $a : \Omega \to \mathbb{R}^d$ on a domain $\Omega \subset \mathbb{R}^d$, the integral operator $\mathcal{G}$ can be formalized as:

$$\mathcal{G}a(\mathbf{x}) = \int_\Omega \kappa(\mathbf{x}, \xi) a(\xi)\, d\xi, \tag{18}$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^d$ and $\kappa(\cdot,\cdot)$ denotes the kernel function defined on Ω. We define the kernel for the graph attention mechanism as follows:

$$\kappa(\mathbf{x}, \xi) = \left(\int_\Omega \exp\left(\mathbf{a}^\top \left[\mathbf{W}a(\xi) \| \mathbf{W}a(\xi')\right]\right) d\xi'\right)^{-1} \exp\left(\mathbf{a}^\top \left[\mathbf{W}a(\mathbf{x}) \| \mathbf{W}a(\xi)\right]\right) \mathbf{W}, \tag{19}$$

where $\mathbf{a} \in \mathbb{R}^{2d}$, $\mathbf{W} \in \mathbb{R}^{d\times d}$ and $\|$ denotes vector concatenation. Using the softmax normalizing function across the domain Ω, assuming Ω carries a uniform measure for simplicity in the Monte-Carlo approximation.

Approximating the integrals by a finite sum over the neighborhood $\mathcal{N}(\mathbf{x})$ of $\mathbf{x}$ in a graph:

$$\int_\Omega \exp\left(\mathbf{a}^\top \left[\mathbf{W}a(\xi) \| \mathbf{W}a(\xi')\right]\right) d\xi' \approx \frac{|\Omega|}{|\mathcal{N}(\mathbf{x})|} \sum_{i \in \mathcal{N}(\mathbf{x})} \exp\left(\mathbf{a}^\top \left[\mathbf{W}a(\mathbf{x}_i) \| \mathbf{W}a(\xi)\right]\right), \tag{20}$$

we then use this to express $\mathcal{G}a(\mathbf{x})$ of Eq.(18) as a weighted sum:

$$\mathcal{G}a(\mathbf{x}) \approx \frac{\sum_{i\in\mathcal{N}(\mathbf{x})} \exp(\mathbf{a}^\top \left[\mathbf{W}a(\mathbf{x}) \| \mathbf{W}a(\mathbf{x}_i)\right]) \mathbf{W}a(\mathbf{x}_i)}{\sum_{j\in\mathcal{N}(\mathbf{x})} \exp(\mathbf{a}^\top \left[\mathbf{W}a(\mathbf{x}) \| \mathbf{W}a(\mathbf{x}_j)\right])}, \tag{21}$$

which is the computation for the graph attention mechanism of GraphFormer defined in Eq.(14)-(15), showing its equivalence to a learnable integral operator on the domain Ω. □

C DETAILS FOR BENCHMARKS

We evaluate AMG across various datasets, including those specifically generated for this study using COMSOL [26].

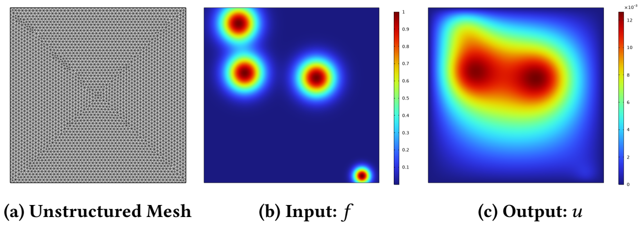

(a) Unstructured Mesh (b) Input: f (c) Output: u

Figure 5: Visualization of a random sample from the Poisson dataset, showing the unstructured mesh, the input source term f and the corresponding solution u.

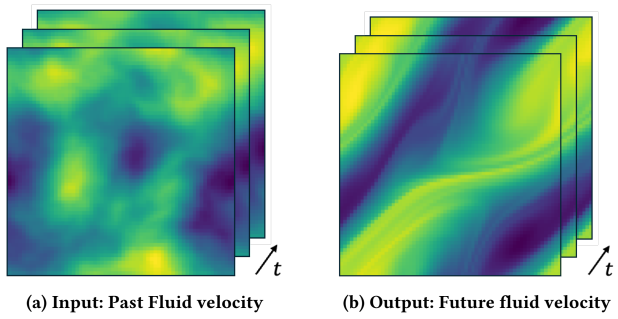
C.1 Poisson Equation

The Poisson equation with Dirichlet boundary conditions is studied:

$$-\Delta u = f, \quad \text{in } \Omega = [0,1]^2, \tag{22}$$

$$u = 0, \quad \text{on } \partial\Omega, \tag{23}$$

where f consists of a Gaussian superposition, with parameters $\mu_{x,i}, \mu_{y,i} \sim \mathrm{U}(0,1)$ and $\sigma_i \sim \mathrm{U}(0.025, 0.1)$. The dataset includes 4000 training, 500 validation, and 500 test samples.



(a) Input: Past Fluid velocity (b) Output: Future fluid velocity

Figure 6: Visualization of the Navier-Stokes dataset for a sample simulation.

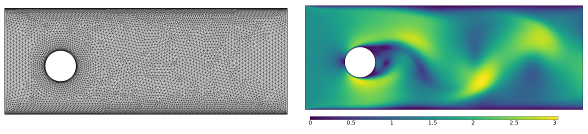
C.2 Navier-Stokes Equation

This section explores the Navier-Stokes equations in vorticity form on a unit torus:

$$\partial_t \omega + \mathbf{u} \cdot \nabla\omega - \nu\Delta\omega = f, \tag{24}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{25}$$

The dataset employs a 64 × 64 grid, with periodic boundary conditions. It comprises 9600 training, 1200 validation, and 1200 test samples.



(a) Mesh of the sample (b) Velocity field in future

Figure 7: Visualization of a sample from the Cylinder Flow.

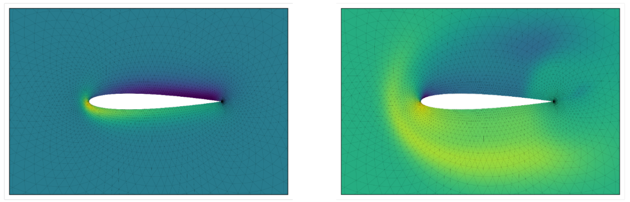
C.3 Cylinder Flow

This dataset captures incompressible fluid dynamics around a 2D circular cylinder within a channel:

$$\nabla \cdot \mathbf{u} = 0, \tag{26}$$

$$\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \nu \nabla^2 \mathbf{u} - \frac{1}{\rho}\nabla p, \tag{27}$$

with boundary conditions set for velocity and pressure. It features 100 snapshots per case, with 7600 training, 1000 validation, and 1000 test samples.



(a) Input: Fluid quantity at the current timestep (b) Output: Predicted fluid quantity at the next timestep

Figure 8: Visualization of the Airfoil dataset illustrating a typical simulation snapshot.

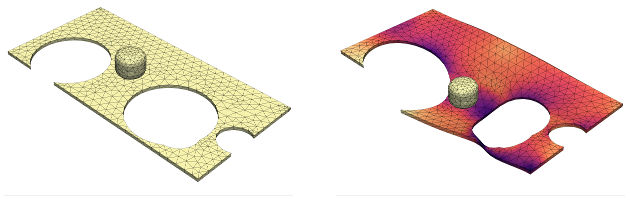
C.4 Airfoil

Simulating subsonic airflow over an airfoil with the 2D compressible Euler equations:

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{28}$$

$$\partial_t(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + p\mathbf{I}) = 0, \tag{29}$$

The simulation uses an unstructured mesh over 10 timesteps. This dataset consists of 10,000 training, 1,000 validation and testing samples each.

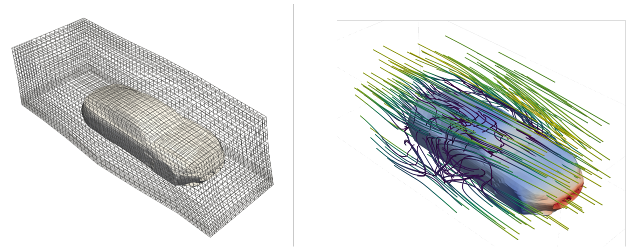


(a) Input: Initial Stress (b) Output: Predicted Stress

Figure 9: Visualization of the input and output of Von Mises stress of a deforming plate.

C.5 Deforming Plate

A 3D dynamic simulation of a deformable plate is modeled in hyperelastic material properties: Data spans 50 timesteps, comprising 24,000 training samples, with 2,000 for validation and testing.



(a) Input: Fluid quantity at the current timestep (b) Output: Predicted fluid quantity at the next timestep

Figure 10: Visualization of the Shape-Net Car dataset.

C.6 Shape-Net Car

Drag coefficient estimation for cars simulated at 72 km/h, with 889 total samples: 690 for training, 99 for validation, and 100 for testing. Predicts velocity and pressure to compute drag coefficients.