GITO: Graph-Informed Transformer Operator for Learning Complex Partial Differential Equations

Milad Ramezankhani [*1] **Janak M. Patel** [*1] **Anirudh Deodhar** [1] **Dagnachew Birru** [1]

Abstract

We present a novel graph-informed transformer operator (GITO) architecture for learning complex partial differential equation systems defined on irregular geometries and non-uniform meshes. GITO consists of two main modules: a hybrid graph transformer (HGT) and a transformer neural operator (TNO). HGT leverages a graph neural network (GNN) to encode local spatial relationships and a transformer to capture long-range dependencies. A self-attention fusion layer integrates the outputs of the GNN and transformer to enable more expressive feature learning on graph-structured data. TNO module employs linear-complexity cross-attention and self-attention layers to map encoded input functions to predictions at arbitrary query locations, ensuring discretization invariance and enabling zero-shot super-resolution across any mesh. Empirical results on benchmark PDE tasks demonstrate that GITO outperforms existing transformer-based neural operators, paving the way for efficient, mesh-agnostic surrogate solvers in engineering applications.

1. Introduction

Solving partial differential equations (PDEs) underpins a vast array of phenomena in engineering and the physical sciences, from fluid flow and heat transfer to fracture mechanics and structural deformation. Traditional numerical methods offer rigorous error bounds and adaptable frameworks, but they often incur substantial computational costs when applied to high-dimensional, nonlinear, or time-dependent problems (Olver et al., 2014). This computational burden can become prohibitive in real-time control and optimization tasks, motivating the search for surrogate models that deliver rapid yet accurate PDE solutions.

In recent years, deep neural network–based surrogates have emerged as a powerful alternative, demonstrating orders-of-magnitude speedups over classical solvers while maintaining competitive accuracy (Zhu & Zabaras, 2018; Bhatnagar et al., 2019). These data-driven models can learn solution operators from precomputed simulation data, enabling instantaneous inference once trained. Physics-informed neural networks (PINNs) (Raissi et al., 2019) introduced a paradigm shift by embedding the governing PDE residual directly into the loss function, thus bypassing the need for labeled solution data. While PINNs have been successfully applied to a wide range of forward and inverse problems, each new setting of initial conditions, boundary values, or forcing terms requires retraining from scratch, constraining their applicability to a single PDE configuration (Chen & Koohy, 2024; Ramezankhani & Milani, 2024).

Neural operators extend the concept of surrogate modeling by directly mapping infinite-dimensional input-output spaces, effectively learning solution operators for a family of PDEs. Foundational architectures such as DeepONet (Lu et al., 2021) and the Fourier Neural Operator (FNO) (Li et al., 2020a) show that a single model can generalize across varying PDE conditions and enable zero-shot super-resolution. Inspired by the success of the transformer architecture (Vaswani et al., 2017) in natural language processing and computer vision, recent works explored attention-based surrogate models to simulate physical systems. Typically, these models are trained on function samples defined over fixed discretization grids, which limits their ability to generalize across varying meshes (Cao, 2021; Han et al., 2022). To address this, a new class of transformer-based neural operators has emerged, which enables super-resolution and discretization-invariant query of the output function (Li et al., 2022b; Hao et al., 2023; Alkin et al., 2024). They employ cross-attention to aggregate input features and predict outputs at arbitrary spatial/temporal coordinates, regardless of the underlying input grid.

Despite these early successes, significant challenges remain in scaling transformer-based operators to realistic engineering applications. In particular, modeling systems with ir-

[*]Equal contribution [1]Applied Research, Quantiphi, Marlborough, MA 01752, USA. Correspondence to: Milad Ramezankhani <milad.ramezankhani@quantiphi.com>, Janak M. Patel <janak.patel@quantiphi.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

regular geometries and non-uniform meshes demands more powerful mechanisms to capture complex interactions and dynamics among spatial nodes. To address these challenges, we propose a novel graph-informed transformer operator (GITO) architecture tailored for mesh-agnostic operator learning on arbitrary domains (Figure 1). Our framework comprises two core modules: a hybrid graph transformer (HGT) and a transformer neural operator (TNO). HGT marries graph neural networks (GNNs) for modeling intricate local spatial relations with transformer layers for long-range, global dependencies, interleaving message-passing and self-attention via a dedicated fusion layer to produce expressive relational embeddings. Building on these embeddings, TNO applies cross-attention for discretization-invariant querying of the output domain, followed by self-attention to capture dependencies among enriched query embeddings. Our main contributions are: 1) a novel graph-transformer-based neural operator architecture that seamlessly integrates local and global feature learning on irregular meshes and geometries, and 2) superior performance on benchmark PDE tasks, outperforming existing transformer-based neural operators.

2. Related work

Transformers as neural operators. The attention mechanism has shown promise at modeling both spatial correlations and temporal dynamics in physical systems. Spatial attention layers aggregate information across nonlocal points, capturing structural patterns and long-range dependencies within the domain (Wu et al., 2024; Hao et al., 2023; Li et al., 2022b; Bryutkin et al., 2024). In the temporal setting, transformers learn state evolution over time without relying on recurrent architectures, often delegating spatial aggregation to other mechanisms such as GNNs (Alkin et al., 2024; Han et al., 2022; Geneva & Zabaras, 2022). In addition, recent work has focused on developing novel transformer architectures to improve the scalability and effectiveness of modeling complex physical systems (Fonseca et al., 2023; Li et al., 2023; Chen & Wu, 2024). Our method captures the spatial structures via linear-complexity attention mechanisms by leveraging the proposed HGT and TNO modules.

Graphs as neural PDE solvers. GNNs have been explored as mesh-agnostic PDE solvers by representing spatial discretizations as graph vertices and leveraging message-passing to model local interactions (Brandstetter et al., 2022; Li et al., 2020b). Previous studies have demonstrated that GNNs can effectively model diverse physical phenomena ranging from fluid dynamics and deformable materials (Sanchez-Gonzalez et al., 2018) to global-scale weather forecasting (Lam et al., 2023). Recently, transformer-inspired architectures have been applied to graph-based operator learning to more effectively handle arbitrary geometries and boundary conditions (Bryutkin et al., 2024). In parallel, latent-space compression via graph encodings has enabled efficient dynamics propagation and scalable temporal rollouts (Alkin et al., 2024; Han et al., 2022).

3. Methodology

3.1. Graph construction and feature encoding

We represent both the input function and query points as separate graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each node $i \in \mathcal{V}$ corresponds to a spatial location (e.g., a mesh cell or a query point) and each edge $(i, j) \in \mathcal{E}$ connects node i either to its k nearest neighbors or to nodes within a specified Euclidean radius. The value of k and radius are considered as model hyperparameters (Appendix B). Each node feature vector V_i includes the spatial coordinates $\mathbf{x}_i$. For nodes corresponding to the input function, the observed field value $\mathbf{u}_i$ is concatenated to the node features. Edge features $\mathbf{E}_{ij}$ comprise relative displacements $(\mathbf{x}_i - \mathbf{x}_j)$, Euclidean distances $|\mathbf{x}_i - \mathbf{x}_j|$, and, in case of input function graphs, differences in solution values between connected nodes $\mathbf{u}_i - \mathbf{u}_j$ (Brandstetter et al., 2022). Both node and edge features are passed through dedicated MLP-based encoders to generate initial embeddings, which are then fed into the HGT layers for subsequent representation learning.

3.2. Hybrid graph transformer (HGT) module

Despite their strengths, GNNs suffer from fundamental limitations due to sparse message passing, notably over-smoothing (Oono & Suzuki, 2019) and over-squashing (Alon & Yahav, 2020). Graph transformers (GTs) (Dwivedi & Bresson, 2020; Ying et al., 2021; Mialon et al., 2021) address these shortcomings by allowing nodes to attend to all others in the graph; however, they often overlook edge features, hindering accurate representation learning. Hybrid architectures such as GPS Graph (Rampášek et al., 2022) and Exphormer (Shirzad et al., 2023) combine GNN and transformer layers to overcome these challenges: the GNN component captures local interactions and integrates edge information, while the transformer module models long-range and global dependencies and mitigates over-smoothing and over-squashing. Following this paradigm, we employ a GNN layer (`GNN`) alongside a linear self-attention module (`GlobalAttn`) to learn graph dynamics and introduce a fusion layer (`Fusion`) that applies self-attention to interleave local neighborhood aggregation with global attention, resulting in richer and more expressive graph representations (Figure 2). In the HGT module, node representations are updated by concatenating the outputs of the `GNN` and `GlobalAttn` layers, followed by processing the combined

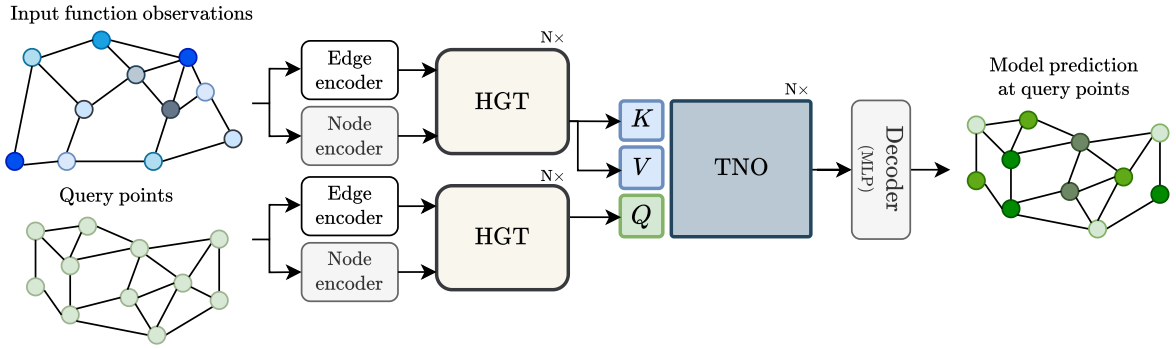

Figure 1. Overall architecture of GITO. The input function and query points are first converted into graph representations and encoded via edge and node encoders. These encoded graphs are then processed by the hybrid graph transformer (HGT) module to learn informative relational features. The output representations from the HGT are used as key/value and query inputs to the transformer neural operator (TNO) module, which integrates contextual information from input function observations to enrich the query representations. Finally, an MLP decoder maps the query embeddings to real spatial coordinates.

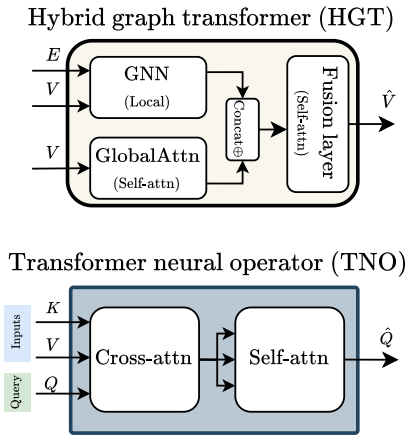

Figure 2. (Top) The hybrid graph transformer (HGT) module consists of a GNN layer, a self-attention global layer, and a self-attention fusion layer that jointly learn graph-based representations. (Bottom) The transformer neural operator (TNO) module employs cross-attention and self-attention mechanisms to integrate and process representations of input functions and query points. For clarity, standard components such as layer normalization, residual connections, and feed-forward networks are omitted.

embedding through the `Fusion` layer:

$$V_G, E = \texttt{GNN}(V, E) \tag{1}$$

$$V_T = \texttt{GlobalAttn}(V) \tag{2}$$

$$\hat{V} = \texttt{Fusion}\left(V_G \oplus V_T\right). \tag{3}$$

The modularity of the hybrid graph transformer enables seamless integration of diverse GNN architectures and transformer modules, allowing the model to be tailored to specific application requirements.

3.3. Transformer neural operator (TNO) module

To empower zero-shot super-resolution and fully decouple input and output representations, we integrate a cross-attention layer capable of querying the output domain at arbitrary spatial locations (Figure 2). This design parallels the branch and trunk networks in the DeepONet (Li et al., 2020b), seamlessly fusing input function embeddings with output queries to achieve discretization-invariant evaluation, regardless of the underlying input mesh (Li et al., 2022b). The cross-attention layer takes as input the query embeddings and the input function representations generated by the HGT modules. The cross-attention enriches the query embeddings with the information from the input functions. A subsequent self-attention module then captures interactions and dependencies among the enriched query points. Finally, an MLP decoder translates the resulting embeddings into the target physical output values.

3.4. Model implementation details

To efficiently learn operators for large-scale physical systems with numerous input and query locations, we adopt the linear-complexity attention mechanism proposed by Hao et al. (2023). Similar to Fourier and Galerkin attention mechanisms (Cao, 2021), this approach can capture complex dynamics while avoiding the quadratic computational cost of softmax-based attention. We adopt a "Norm-Attn-MLP-Norm" with residual connections for all attention layers. To handle cases with multiple input functions, we use a dedicated encoder for each input function. These encoded representations are then processed by the cross-attention module in TNO, specifically designed to handle multiple key-value (K/V) combinations, enabling efficient interaction across heterogeneous inputs. We incorporate a mixture of experts module following each attention mechanism. The

Dataset	Subset	MIONet	FNO	GKT	Geo-FNO	OFormer	GNOT	**GITO (Ours)**	Improvement
NS	u	2.74e-2	6.56e-2	1.52e-2	1.41e-2	2.33e-2	<u>1.05e-2</u>	**8.19e-3**	22 %
	v	5.51e-2	1.15e-1	3.15e-2	2.98e-2	4.83e-2	<u>2.33e-2</u>	**2.02e-2**	13.3 %
	p	2.74e-2	<u>1.11e-2</u>	1.59e-2	1.62e-2	2.43e-2	1.23e-2	**1.07e-2**	3.6 %
Heat	T	1.74e-1	–	–	–	–	<u>5.42e-2</u>	**4.49e-2**	17.2 %
Elasticity	σ	9.65e-2	5.08e-2	2.01e-2	2.20e-2	1.83e-2	<u>9.04e-3</u>	**8.87e-3**	1.9%

Table 1. Comparison of GITO with existing operator learning methods on the NS, Heat and Elasticity datasets. The metric used for this comparison is relative L^2 error, with lower scores indicating better performance. The top **first** and <u>second</u> best results are highlighted. Results for all models except GNOT and GITO are taken directly from Hao et al. (2023). For a fair comparison, we trained a smaller GNOT model to match GITO's model size (see Appendix A for details.)

gating network assigns weights to the experts based on the spatial location of the query points, effectively promoting a form of *soft* domain decomposition, which has been shown to enhance the learning of physical systems in prior work (Chalapathi et al., 2024; Hao et al., 2023). In the HGT module, we use the Graph Attention Network (GATv2) (Brody et al., 2021) as the GNN layer and apply the same linear-complexity attention mechanism as in TNO for both the global and fusion layers. The graph construction strategies are detailed in Appendix B. In this work, we choose to use the HGT module only for query points for learning more expressive relational features.

4. Experimental results

Datasets. To demonstrate the scalability and effectiveness of GITO on complex geometries, we evaluate it on three challenging datasets based on Navier–Stokes, heat conduction, and elasticity (Li et al., 2022a; Hao et al., 2023), which govern critical physical processes in buildings and structural engineering. The Navier–Stokes dataset models incompressible flow, relevant to wind loading, ventilation, and air leakage; the heat conduction dataset captures temperature profiles in multi-material systems, key for assessing thermal insulation, thermal bridging, and the energy performance of building envelopes; and the elasticity dataset simulates stress-strain responses under mechanical loading, essential for evaluating structural stability, deformation, and material performance. A brief overview of the datasets is provided below, with detailed descriptions available in Appendix A.

1) 2D steady-state Navier-Stokes (NS): This dataset involves steady 2D fluid flow governed by Navier-Stokes equations in a rectangular domain with varying cavity positions (Figure 3.a). The goal is to predict velocity components u, v, and pressure p from the input mesh; **2) Multilayer 2D Heat Conduction (Heat)**: This dataset models heat conduction in composite media with multiple boundary shapes and spatially varying boundary conditions (Figure 3.b). The task is to predict temperature T from multiple input functions; **3) Elasticity**: This dataset simulates solid mechanics governed by elastokinetics equations. The domain is a unit square containing an irregular cavity, and the objective is to predict the stress field given the input mesh.

Baseline Models. We benchmark our model against both conventional neural operator architectures, including FNO (Li et al., 2020a), Geo-FNO (Li et al., 2022a), and MIONet (Jin et al., 2022), as well as recently developed transformer-based operators, namely, GNOT (Hao et al., 2023), Galerkin Transformer (GKT) (Cao, 2021), and OFormer (Li et al., 2022b). GNOT has demonstrated state-of-the-art performance across various PDE benchmarks. To ensure a fair comparison, we re-implement GNOT and evaluate it under the same experimental settings as our model, using a comparable or slightly larger number of parameters (Appendix A). We directly report the performance of other baseline models from Hao et al. (2023).

Results. Table 1 summarizes the mean relative L^2 error on each test dataset for all compared models, where lower values denote higher accuracy. Detailed hyperparameter configurations appear in Appendix A. Across every benchmark, GITO consistently outperforms the baselines, achieving the lowest error against the nearest competitor (GNOT) while employing a similar or slightly reduced parameter count. Although FNO demonstrates marginally better performance than GNOT on the p variable of the NS dataset, the proposed GITO model surpasses FNO in this regard as well. GITO also achieves the lowest error on the Elasticity dataset, indicating its robustness even in highly sparse and irregular domains. Consequently, GITO maintains superior overall accuracy across all tasks. These findings demonstrate GITO's generality and efficacy in handling both complex geometries (NS and Elasticiy dataset, Figure 4) and multi-input settings (Heat dataset), establishing it as a versatile, high-performance surrogate for diverse scientific and engineering applications.

We further conducted ablation studies to assess the impact of two critical design components in GITO: the fusion layer in the HGT module and the graph construction strategy. To evaluate the contribution of the fusion layer, we conducted

Configuration→		GITO w/ Fusion	GITO w/o Fusion
Relative L^2 Error	u	8.42e-3	1.00e-2
	v	2.06e-2	2.44e-2
	p	1.12e-2	1.61e-2
Model Parameters (M)		4.75	5.35

Table 2. Ablation study comparing GITO with and without the fusion layer on the NS dataset. The fusion layer combines outputs from the GNN and self-attention paths. Reported values are relative L^2 errors; lower is better.

experiments on the NS dataset using identical hyperparameters, except for the hidden size. In this variant, the outputs of the GNN and self-attention modules were summed and passed through a 2-layer MLP (as in the GPS Graph design (Rampášek et al., 2022)), instead of being concatenated and fused. To match the dimensionality of the original fused output, the hidden size of the non-fusion model was doubled (192 vs. 96). Table 2 reports the number of model parameters and the relative L^2 errors for both configurations. Despite having more parameters, the model without the fusion layer exhibited consistently worse accuracy across all predicted variables. This clearly demonstrates the effectiveness of the fusion mechanism in enabling more expressive feature interactions between the GNN and self-attention pathways, as opposed to the limited representational capacity of a simple element-wise summation. The results for different graph construction methods are presented in Appendix B.1 (Tables 4 and 5).

5. Conclusion

In this work, we introduced GITO, the Graph-Informed Transformer Operator, a novel architecture that unifies graph neural networks with transformer attention to learn mesh-agnostic PDE solution operators for arbitrary geometries. By combining hybrid message-passing, discretization-invariant cross-attention, and scalable linear-complexity attention mechanisms, GITO delivers zero-shot super-resolution and outperforms existing transformer-based operators across diverse benchmarks. These results underscore GITO's promise as an accurate and efficient surrogate model for complex engineering applications.

References

Alkin, B., Fürst, A., Schmid, S., Gruber, L., Holzleitner, M., and Brandstetter, J. Universal physics transformers: A framework for efficiently scaling neural operators. *Advances in Neural Information Processing Systems*, 37: 25152–25194, 2024.

Alon, U. and Yahav, E. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., and Kaushik, S. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64:525–545, 2019.

Brandstetter, J., Worrall, D., and Welling, M. Message passing neural pde solvers. *arXiv preprint arXiv:2202.03376*, 2022.

Brody, S., Alon, U., and Yahav, E. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

Bryutkin, A., Huang, J., Deng, Z., Yang, G., Schönlieb, C.-B., and Aviles-Rivero, A. Hamlet: Graph transformer neural operator for partial differential equations. *arXiv preprint arXiv:2402.03541*, 2024.

Cao, S. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34: 24924–24940, 2021.

Chalapathi, N., Du, Y., and Krishnapriyan, A. Scaling physics-informed hard constraints with mixture-of-experts. *arXiv preprint arXiv:2402.13412*, 2024.

Chen, J. and Wu, K. Positional knowledge is all you need: Position-induced transformer (pit) for operator learning. *arXiv preprint arXiv:2405.09285*, 2024.

Chen, Y. and Koohy, S. Gpt-pinn: Generative pre-trained physics-informed neural networks toward non-intrusive meta-learning of parametric pdes. *Finite Elements in Analysis and Design*, 228:104047, 2024.

Dwivedi, V. P. and Bresson, X. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

Fonseca, A. H. d. O., Zappala, E., Caro, J. O., and Van Dijk, D. Continuous spatiotemporal transformers. *arXiv preprint arXiv:2301.13338*, 2023.

Geneva, N. and Zabaras, N. Transformers for modeling physical systems. *Neural Networks*, 146:272–289, 2022.

Han, X., Gao, H., Pfaff, T., Wang, J.-X., and Liu, L.-P. Predicting physics in mesh-reduced space with temporal attention. *arXiv preprint arXiv:2201.09113*, 2022.

Hao, Z., Wang, Z., Su, H., Ying, C., Dong, Y., Liu, S., Cheng, Z., Song, J., and Zhu, J. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pp. 12556–12569. PMLR, 2023.

Jin, P., Meng, S., and Lu, L. Mionet: Learning multiple-input operators via tensor product. *arXiv preprint arXiv:2202.06137*, 2022.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

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

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

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

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022b.

Li, Z., Shu, D., and Barati Farimani, A. Scalable transformer for pde surrogate modeling. *Advances in Neural Information Processing Systems*, 36:28010–28039, 2023.

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

Mialon, G., Chen, D., Selosse, M., and Mairal, J. Graphit: Encoding graph structure in transformers. *arXiv preprint arXiv:2106.05667*, 2021.

Olver, P. J. et al. *Introduction to partial differential equations*, volume 1. Springer, 2014.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Ramezankhani, M. and Milani, A. S. A sequential meta-transfer (smt) learning to combat complexities of physics-informed neural networks: Application to composites autoclave processing. *Composites Part B: Engineering*, 283:111597, 2024.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. In *International conference on machine learning*, pp. 4470–4479. PMLR, 2018.

Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D. J., and Sinop, A. K. Exphormer: Sparse transformers for graphs. In *International Conference on Machine Learning*, pp. 31613–31632. PMLR, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wu, H., Luo, H., Wang, H., Wang, J., and Long, M. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do transformers really perform badly for graph representation? *Advances in neural information processing systems*, 34:28877–28888, 2021.

Zhu, Y. and Zabaras, N. Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification. *Journal of Computational Physics*, 366:415–447, 2018.

A. Datasets and Model Hyperparameters.

Datasets.

NS. We use a two-dimensional steady-state fluid dynamics dataset governed by the incompressible Navier–Stokes equations. The computational domain is a square region of size $[0, 8]^2$ with four internal circular cavities, resulting in a complex and non-trivial geometry. The goal is to predict the velocity components in the x and y directions, denoted by u and v, respectively, along with the pressure field p, given the input mesh geometry. The domain is defined as

$$\Omega = [0, 8]^2 \setminus \bigcup_{i=1}^{4} R_i \tag{1}$$

where each R_i represents a circular cavity. The flow within this domain is governed by the steady-state incompressible Navier–Stokes equations:

$$(\mathbf{u} \cdot \nabla)\mathbf{u} = \frac{1}{\mathrm{Re}} \nabla^2 \mathbf{u} - \nabla p \tag{2}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3}$$

where $\mathbf{u} = (u, v)$ is the velocity vector field and Re is the Reynolds number. The boundary conditions are defined as follows: the velocity is set to zero on the entire boundary, i.e., $\mathbf{u} = 0$ on $\partial\Omega$. A parabolic velocity profile is imposed at the inlet (left boundary), given by $u_x = \frac{y(8-y)}{16}$, while at the outlet (right boundary), the pressure is fixed at $p = 0$.

Each sample in the dataset consists of a two-dimensional unstructured mesh over the domain Ω and the corresponding geometric configuration determined by the locations of the circular cavities. The outputs are the velocity field (u, v) and the pressure field p, evaluated over the mesh nodes. We use the dataset provided by Hao et al. (2023), which is publicly available at `https://github.com/HaoZhongkai/GNOT`. It consists of 1,100 samples generated by varying the positions of the four cavities to create different internal geometries. Of these, 1,000 samples are used for training and 100 for testing.

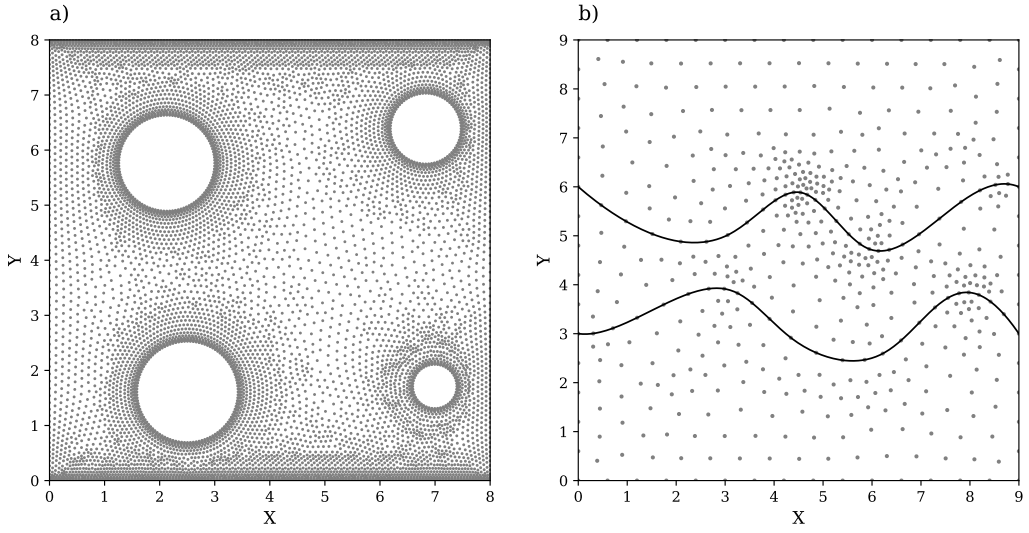

Figure 3. Visualization of mesh points depicting the domain geometry of a single sample from a) the NS dataset and b) the Heat dataset, where the geometry varies across samples.

Heat. We use a multi-scale heat conduction dataset, where the goal is to predict the steady-state temperature field T given spatially varying material properties and complex boundary conditions. The problem is governed by the two-dimensional steady-state heat equation:

$$\rho C_p \mathbf{u} \cdot \nabla T - k \nabla^2 T = Q \tag{4}$$

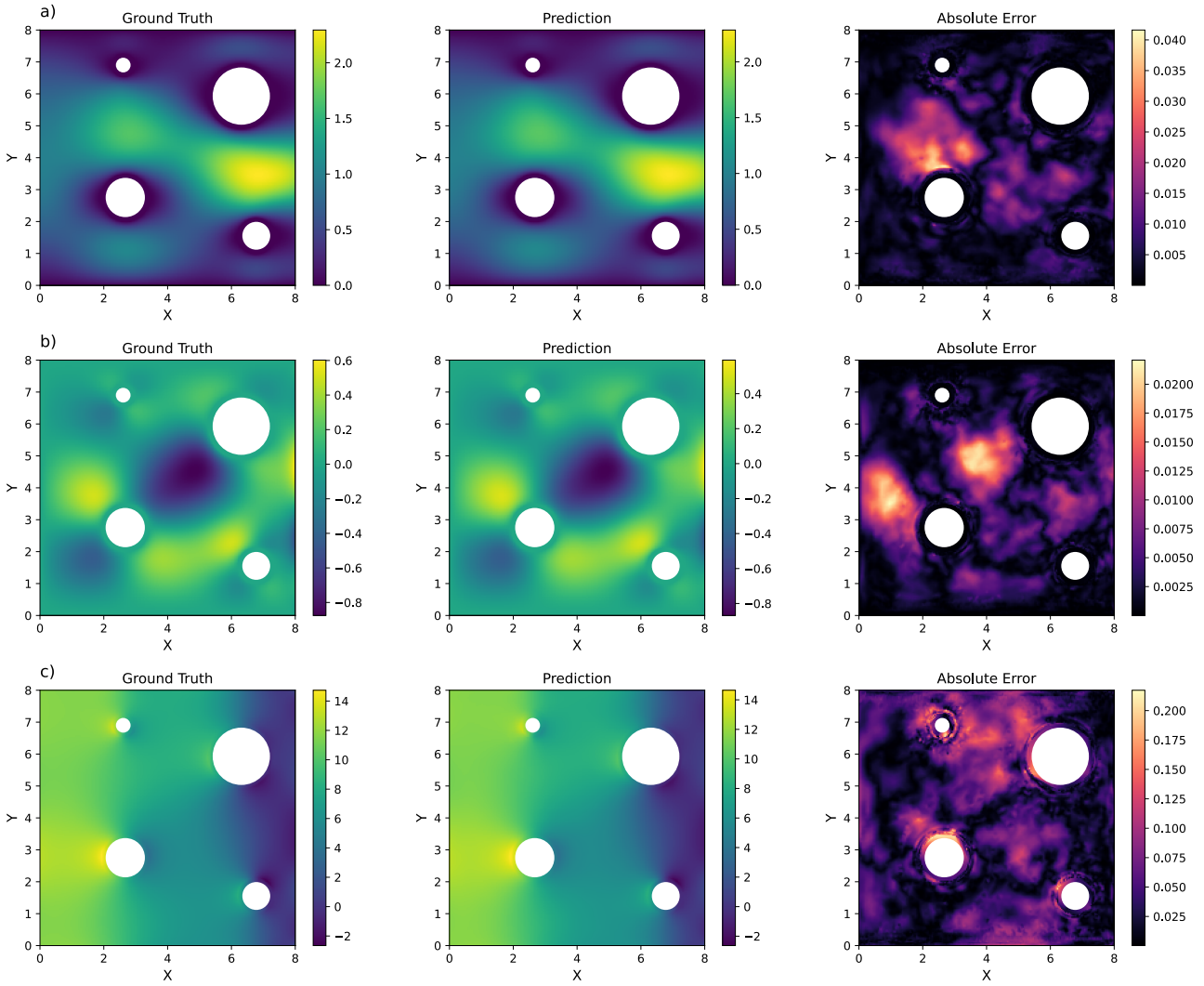

Figure 4. Comparison of GITO's predictions against ground truth, and the corresponding absolute error plots for a test sample from the NS dataset: a) velocity component u, b) velocity component v, and c) pressure p.

where ρ is the density, C_p is the specific heat capacity, $\mathbf{u}$ is the velocity field, k is the thermal conductivity, and Q is the internal heat source. The computational domain is a rectangle defined as $\Omega = [0, 9]^2$, segmented into three subdomains using two spline-shaped curves. Each subdomain possesses distinct physical properties, resulting in a highly heterogeneous medium. Periodic boundary conditions are applied along the left and right edges of the domain. The top boundary is assigned a temperature profile, which acts as an input function. Another input function parameterizes the shapes of the internal splines, thereby influencing the geometry of the subdomains and the overall distribution of material properties.

Each sample in the dataset includes a two-dimensional unstructured mesh over the domain Ω and five input functions: the geometric configuration defined by the spline parameters and the temperature distribution prescribed on the top boundary. The output is the temperature field T evaluated over the same discretized mesh. We use the dataset provided by Hao et al. (2023), which is publicly available at `https://github.com/HaoZhongkai/GNOT`. It consists of 1,100 samples generated by varying the spline parameters and boundary conditions to produce diverse thermal configurations. Of these, 1,000 samples are used for training and 100 for testing.

Elasticity. We use a two-dimensional elasticity dataset governed by elastokinetics equations, modeling the mechanical response of a solid body under external loading. The computational domain is a unit square $\Omega = [0, 1]^2$ with a centrally

Hyperparameter Type	NS		Heat		Elasticity	
	GNOT	GITO	GNOT	GITO	GNOT	GITO
Activation function	GELU	GELU	GELU	GELU	GELU	GELU
Number of attention layers	2	2	3	3	2	2
Hidden size of attention	192	96	256	128	192	96
Layers of MLP	2	2	4	3	2	2
Hidden size of MLP	192	96	256	128	192	96
Hidden size of input embedding	192	96	256	128	192	96
Learning rate schedule	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle	OneCycle
N experts	2	2	4	3	2	2
N heads	8	8	8	8	8	8
Total model parameters	4.49M	4.37M	21.26M	18.24M	4.48M	4.35M

Table 3. Hyperparameters and training runtime details for the GNOT and GITO models on the NS, Heat and Elasticity datasets. Hidden sizes for GITO are reported before fusion.

located irregular cavity, resulting in complex internal geometry. The objective is to predict the stress field components over the domain, given the input mesh geometry. The governing equation for the displacement field u in a solid body is given by:

$$\rho^s \frac{\partial^2 u}{\partial t^2} + \nabla \cdot \sigma = 0 \tag{5}$$

where ρ^s is the mass density, and σ is the stress tensor.

Each sample in the dataset consists of a two-dimensional point cloud discretization of the domain, along with the corresponding cavity geometry. The inputs are the point cloud coordinates, and the outputs are the stress field components evaluated at each point. We use the dataset provided by Li et al. (2022a), which is publicly available at `https://github.com/neuraloperator/Geo-FNO`. It comprises over 2,000 samples generated by varying the shape and location of the internal cavity to create diverse geometric configurations. Of these, 1,000 samples are used for training and 200 for testing.

Model Hyperparameters. Table 3 outlines the hyperparameters used in our experiments for all benchmarks across the GNOT and GITO models. To ensure a fair comparison, we kept the number of attention layers, MLP layers, and experts identical across both models. However, since GITO employs a fusion layer that concatenates the outputs of the GNN and self-attention components within the HGT module (effectively doubling the hidden dimension) we set the hidden size in GNOT to be twice that of GITO prior to this fusion. Note that the hidden sizes reported in the table for GITO correspond to the dimensions before concatenation. For the Heat dataset, which involves larger inputs and more complex dynamics, we increased the number of MLP layers and experts in GNOT to match or slightly exceed the total parameter count of GITO. Although exact parameter matching was not possible due to architectural differences, we intentionally allocated slightly more parameters to GNOT to fairly showcase its performance and demonstrate its modeling efficacy under comparable capacity constraints. All experiments were conducted on a single NVIDIA V100 GPU with 8 vCPUs and 52 GB of RAM.

B. Additional results

Figure 4 shows a qualitative comparison between our model's predictions and the ground truth for a test sample from the NS dataset. Panels a) and b) display the velocity components u and v, respectively, while panel c) shows the pressure p. The absolute error plots illustrate the spatial distribution and magnitude of prediction errors, highlighting the model's accuracy in capturing complex flow dynamics across all variables.

B.1. Effect of graph construction strategies

We conduct an ablation study to evaluate the impact of graph construction methods on the accuracy and computational efficiency of the proposed GITO model. Specifically, we compare two widely used strategies: **K-nearest neighbors (KNN)** and **radius-based (circular)** graphs (Qi et al., 2017; Bryutkin et al., 2024). In the KNN strategy, each node is connected to its k nearest neighbors based on spatial proximity. In the circular strategy, nodes are connected to all other nodes within a

fixed radius, forming edges only if the pairwise distance falls below the specified threshold.

NS Dataset: As shown in Table 4, increasing the number of neighbors in KNN (from 4 to 16) consistently reduces the relative L^2 error, indicating that denser local connectivity enables better modeling of spatial dependencies. However, this improvement comes at the cost of a significant increase in the number of graph edges (from 41k to 164k), which leads to higher memory usage and computational time during training and inference. On the other hand, the circular strategy with a radius of 0.0525 achieves the lowest error (3.91e-2) while maintaining a moderate edge count (16k). This suggests that, with careful tuning, the radius-based approach can capture only the most relevant local interactions and avoid unnecessary edges, providing a better balance between accuracy and efficiency. Larger radius, such as 0.067, may include irrelevant distant nodes, while smaller radius (e.g., 0.04) risk missing essential local connections—both resulting in slightly degraded performance.

Strategy for graph construction	**KNN (Number of neighbors)**			**Circular (Radius)**		
	4	**8**	**16**	**0.04**	**0.0525**	**0.067**
Relative L^2 error (total)	4.20e-2	4.33e-3	4.17e-2	4.06e-2	3.91e-2	3.93e-2
Number of Edges	41k	82k	164k	8k	16k	44k

Table 4. Ablation study on the effect of graph construction strategies-KNN (with varying number of neighbors) and Circular (with varying radius)-on model accuracy for the NS dataset. The table reports the sum of relative L^2 errors across all three variables (lower is better). The number of edges is reported approximately in thousands.

Heat Dataset: The trends differ for the Heat dataset (Table 5), which features more sparsely distributed query points and different material properties. Here, increasing the KNN count beyond 8 does not yield consistent improvements and, in fact, degrades performance. For instance, using 16 neighbors increases the error to 4.75e-2 compared to 4.61e-2 with 8 neighbors and 4.49e-2 with only 4 neighbors. This is likely because higher KNN values may force connections to spatially distant and physically irrelevant nodes, misleading the model in heterogeneous material settings. Similarly, larger radius in the circular graph (e.g., 0.9) also lead to performance drops due to excessive inclusion of unrelated nodes. The best performance is observed with a small radius (0.25), which maintains sparse yet contextually meaningful connectivity. These results emphasize the importance of tailoring graph construction strategies to the underlying spatial structure and physical properties of the dataset.

Strategy for graph construction	**KNN (Number of neighbors)**			**Circular (Radius)**		
	4	**8**	**16**	**0.25**	**0.4**	**0.9**
Relative L^2 error	4.49e-2	4.61e-2	4.75e-2	4.64e-2	4.72e-2	4.75e-2
Number of Edges	8k	16k	33k	2.5k	8k	36k

Table 5. Ablation study on the effect of graph construction strategies on model performance for temperature prediction in the Heat dataset. Reported values are relative L^2 errors; lower is better. The number of edges is reported approximately in thousands.

Elasticity Dataset. For the Elasticity dataset, we use a radius-based graph construction with a radius of 0.05 to handle the sparse point cloud structure. The chosen radius strikes a balance between maintaining meaningful local connectivity and keeping the graph sparse, leading to improved efficiency and accurate stress field prediction.

Overall, the ablation studies demonstrate that the choice of graph construction strategy significantly affects both the accuracy and computational efficiency of the model. While KNN provides a simple and adaptive structure, circular graphs—when carefully tuned—offer a more interpretable and controllable connectivity pattern, often yielding better performance with fewer edges. For datasets with dense spatial coverage (like NS), moderate-radius circular graphs are preferable, while for sparse or heterogeneous domains (like Heat), lower connectivity thresholds help prevent overfitting to irrelevant neighbors. Ultimately, the best graph construction strategy varies depending on the specific characteristics of the problem domain.