Geometry Aware Operator Transformer As An Efficient And Accurate Neural Surrogate For PDEs On Arbitrary Domains

Shizheng Wen[1] **Arsh Kumbhat**[1] **Levi Lingsch**[1,2]

Sepehr Mousavi[1,3] **Yizhou Zhao**[4] **Praveen Chandrashekar**[5] **Siddhartha Mishra**[1,2]

[1] Seminar for Applied Mathematics, ETH Zurich, Switzerland
[2] ETH AI Center, Zurich, Switzerland
[3] Department of Mechanical and Process Engineering, ETH Zurich, Switzerland
[4] School of Computer Science, CMU, USA
[5] Centre for Applicable Mathematics, TIFR, India

Abstract

The very challenging task of learning solution operators of PDEs on arbitrary domains accurately and efficiently is of vital importance to engineering and industrial simulations. Despite the existence of many operator learning algorithms to approximate such PDEs, we find that accurate models are not necessarily computationally efficient and vice versa. We address this issue by proposing a geometry aware operator transformer (GAOT) for learning PDEs on arbitrary domains. GAOT combines novel multiscale attentional graph neural operator encoders and decoders, together with geometry embeddings and (vision) transformer processors to accurately map information about the domain and the inputs into a robust approximation of the PDE solution. Multiple innovations in the implementation of GAOT also ensure computational efficiency and scalability. We demonstrate this significant gain in both accuracy and efficiency of GAOT over several baselines on a large number of learning tasks from a diverse set of PDEs, including achieving state of the art performance on three large scale three-dimensional industrial CFD datasets. Our project page for accessing the source code is available at camlab-ethz.github.io/GAOT.

1 Introduction

Partial Differential Equations (PDEs) are widely used to mathematically model very diverse natural and engineering systems [15]. Currently, numerical algorithms, such as the finite element and finite difference methods, are the preferred framework for *simulating* PDEs [44]. However, these methods can be computationally very expensive, particularly for the so-called *many-query* problems such as uncertainty quantification (UQ), control, and inverse problems. Here, the solver must be called repeatedly, leading to prohibitive costs and providing the impetus for the design of fast and efficient surrogates for PDE solvers [36].

To this end, ML/AI based algorithms are increasingly being explored as *neural PDE surrogates*. In particular, *neural operators* [23, 5], including those proposed in [26, 27, 32, 45, 21], which learn the *PDE solution operator* from data, are widely used [3]. As many of these neural operators are restricted to PDEs on Cartesian (regular) grids, they cannot be directly applied to most engineering

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

and industrial systems, which are set on domains with complex geometries, imposing a pressing need for neural operators for learning PDEs on arbitrary domains (point clouds).

In this context, a variety of options have recently been proposed, including domain masking for FNO and CNO [45], replacing FFT in FNO with direct spectral evaluations [30], augmenting FNO with learned diffeomorphisms [25] and mapping arbitrary point cloud data between the input domain and latent regular grids with learned encoders/decoders, while processing on the latent grid with FNO [28] or transformers [1, 53]. Alternatives such as end-to-end message-passing based graph neural networks [41, 16, 47, 46, 7, 14, 39] or end-to-end transformer based models [52, 33, 19, 51, 48] have also been proposed. A thorough comparison of the existing models, not just in terms of accuracy but also computational efficiency and scalability, is necessary to evaluate whether these models are yet capable enough to act as surrogates for highly successful finite element methods for engineering simulations [44]. As one of the contributions in this paper, we performed such a comparison (see Sec.3 for details) to find evidence for an *accuracy-efficiency trade-off*, i.e., accurate and robust models, such as the message passing based RIGNO of [39] are not necessarily computationally efficient nor scalable in terms of training throughput and inference latency. On the other hand, more efficient models such as GINO [28] are not accurate enough (see the accompanying Radar Chart in Fig. 1). Given this observation, our main goal in this paper is to propose an algorithm to learn PDE solution operators on arbitrary domains that is accurate, computationally efficient, and can be seamlessly scaled to real-world industrial simulations.

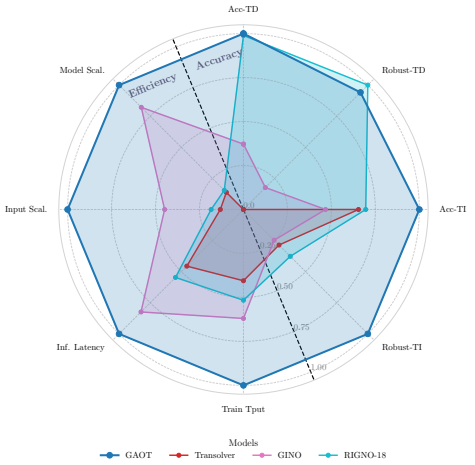


Figure 1: Normalized performance of GAOT and baselines across eight axes, covering accuracy (Acc.), robustness (Robust), throughput (Tput), scalability (Scal.) on time-dependent (TD) and time-independent (TI) tasks.

To this end, we propose *Geometry Aware Operator Transformer* (GAOT, pronounced goat) as a neural surrogate for PDEs on arbitrary domains. While being based on the well-established *encode-process-decode* paradigm [41], GAOT includes several novel features that are designed to ensure both computational efficiency and accuracy, such as

- Our proposed *multiscale attentional graph neural operator* (MAGNO) as the encoder between inputs on an arbitrary point cloud and a *coarser* latent grid, designed to enhance accuracy through its multiscale information processing and attention modules.
- Novel *Geometry embeddings* in the encoder (and decoder) that provide the model with access to information about the (local) domain geometry, greatly increasing accuracy.
- A *transformer processor* that utilizes patching (as in ViTs [11]) for computational efficiency.
- A MAGNO decoder, able to generate *neural fields*, with the ability to approximate the underlying solution at *any query point* in the domain.
- A set of novel implementation strategies to ensure that the computational realization of GAOT is efficient and highly scalable.

Combining these elements allows GAOT to treat PDEs on arbitrary domains in a robust, accurate and computationally efficient manner. We demonstrate these capabilities by,

- Extensively testing GAOT on 28 challenging benchmarks for both time-independent and time-dependent PDEs of various types, ranging from regular grids to random point clouds to highly unstructured adapted grids, and comparing it with widely-used baselines to show that GAOT is both highly accurate as well as computationally efficient and scalable, see Fig. 1.
- The efficiency and scalability of GAOT is further showcased by it achieving state of the art (SOTA) performance on the large scale three-dimensional industrial benchmark of *DrivAerNet++* dataset for automobile aerodynamics [13]. We also test GAOT and demonstrate its superior performance to the GINO baseline on two further 3D industrial scale datasets, i.e.,

DrivaerML dataset for automobile aerodynamics and a NASA-CRM dataset for aerospace applications.

- Through extensive ablations, we also highlight how the novel elements in the design of GAOT such as multiscale attentional encoders and geometry embeddings crucially contribute to the overall performance of our model.

2 Methods.

Problem Formulation. We start with a generic *time-independent PDE*,

$$\mathcal{D}(c,u) = f, \quad \forall x \in D \subset \mathbb{R}^d, \quad \mathcal{B}(u) = u_b, \quad x \in \partial D, \tag{1}$$

with $u : D \mapsto \mathbb{R}^m$, the PDE solution, c is the coefficient (PDE parameters), f is the forcing term, u_b are boundary values and $\mathcal{D}$ and $\mathcal{B}$ are the underlying differential and boundary operators, respectively. Denoting as χ_D, a function (e.g. indicator or signed distance) parameterizing the domain D, we combine all the *inputs* to the PDE (1) together into $a = (c, f, u_b, \chi_D)$, then the *solution operator* $\mathcal{S}$ maps inputs into PDE solution with $u = \mathcal{S}a$. The corresponding *operator learning task* is to learn the solution operator $\mathcal{S}$ from data. To this end, let μ be an underlying *data distribution*. We sample i.i.d. inputs $a^{(i)} \sim \mu$, for $1 \leq i \leq M$ and assume that we have access to *data pairs* $\left(a^{(i)}, u^{(i)}\right)$ with $u^{(i)} = \mathcal{S}a^{(i)}$, Thus, the operator learning task is to approximate the distribution $\mathcal{S}_{\#\mu}$ from these data pairs. In practice, we can only access *discretized* versions of the data pairs, sampled on collocation points (which can vary over samples).

Similarly denoting a generic *time-dependent PDE* as,

$$u_t + \mathcal{D}(c,u) = 0, \quad \forall x \in D \subset \mathbb{R}^d,\ t \in [0,T] \quad u(0) = u_0, \quad x \in D, \tag{2}$$

with, $u : D \times [0,T] \mapsto \mathbb{R}^m$, c the PDE coefficient and u_0 the initial datum and the underlying (spatial) differential operator $\mathcal{D}$. Clubbing the *inputs* to the PDE (2) into $a = (c, u_0, \chi_D)$, the corresponding *solution operator* $\mathcal{S}_t$, with $u(t) = \mathcal{S}_t(a)$ for all $t \in [0,T]$, maps the input into trajectory of the solution. The *operator learning task* consists of approximating $(\mathcal{S}_t)_{\#\mu}$ from data pairs $\left(a^{(i)}, u^{(i)}(t)\right)$ for all $t \in [0, T^{(i)})]$ and $1 \leq i \leq M$ with samples a_i drawn from the data distribution μ. However in practice, we only have access to data, sampled on a discrete set of spatial points per sample as well as only on discrete time snapshots $t_n^{(i)} \in [0, T^{(i)}]$ and have to learn the solution operator from them.

GAOT Model Architecture. The overall architecture of GAOT is depicted in Fig. 2. For simplicity of exposition, we start with the time-independent case, where given inputs $a(x_j)$ on the point cloud $D_\Delta = \{x_j\} \subset D$, for $1 \leq j \leq J$, GAOT provides an approximation to solution u of the PDE (1) at any query point $x \in D$. At a high level, GAOT follows the *encode-process-decode* paradigm of [41]. In the first step, an *encoder* transforms the input on the underlying point cloud D_Δ to a *latent point cloud* $\mathcal{D} \subset \mathbb{R}^d$. The resulting *spatial tokens* are then processed by a processor module to learn useful representations and its output is remapped to the original domain D via the *decoder*, which allows evaluation at any query point $x \in D$.

Choice of Latent Domain. As depicted in **SM** Fig. B.1, the latent domain $\mathcal{D}$ (to which the encoder maps) can be chosen in three different ways, namely i) a regular (structured) grid stencil, consisting of equispaced points on a Cartesian domain (see also Fig. 2) ii) randomly downsampling the underlying point cloud D_Δ or iii) a projected low-dimensional representation, where a high-dimensional domain is projected to a lower dimension (for instance using tri-plane embeddings in 3-D [9]) and a regular grid is used in the lower-dimensional domain. GAOT is a general framework where any of these latent point cloud choices can be employed for $\mathcal{D}$.

Encoder. Given input values $a(x_j)$ on the underlying point cloud D_Δ, the encoder aims to transform it into latent features $w_e(y)$ at any point $y \in \mathcal{D}$ on the latent point cloud. Using a graph-neural operator (GNO) encoder as in GINO [28] would lead to,

$$\widetilde{w}_e(y) = \sum_{k=1}^{n_y} \alpha_k K(y, x_k, a(x_k))\varphi(a(x_k)), \tag{3}$$

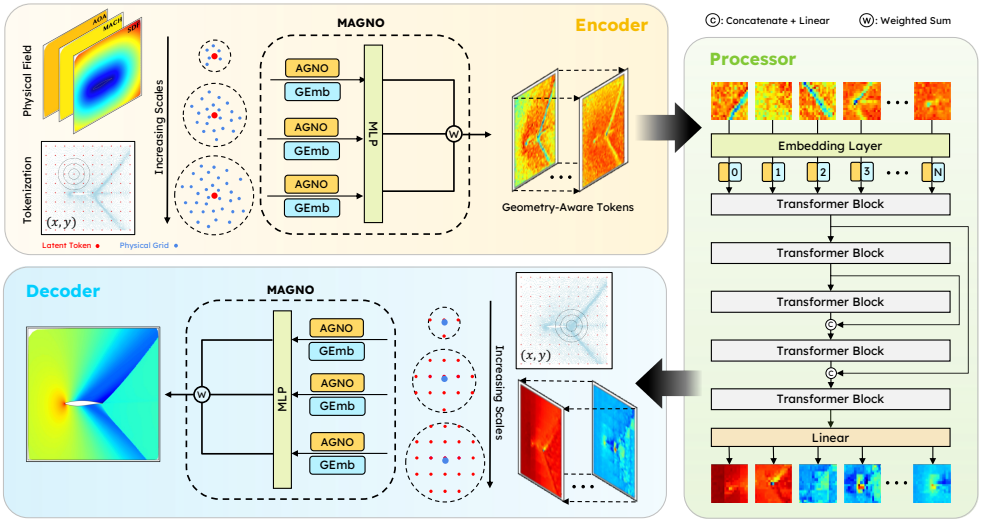


Figure 2: Schematic of the GAOT with an equispaced latent token grid. The encoder uses a multiscale attentional graph neural operator (MAGNO) to aggregate the input data into geometry-aware tokens. A vision transformer (ViT) block with residual connections processes tokens, enabling global exchange of information. A MAGNO decoder identifies the nearest tokens around a given query point to decode the final field.

with MLPs K and φ and the sum above taken over all the n_y points $x_k \in D_\Delta$ such that $|y - x_k| \leq r$ for some hyperparameter $r > 0$, where α_k are some given quadrature weights. In other words, a GNO accumulates information from all the points in the original point cloud that lie inside a ball of radius r, centered at the given point y in the latent point cloud, and processes them through a kernel integral.

Our first innovation is based on the realization that this *single-scale* approach might be limiting the overall accuracy. Instead, we would like to introduce a mechanism to integrate *multiscale* information into the encoder. To this end and as shown in Fig. 2, we choose $r_m = s_m r_0$, for some base radius r_0 and scale factors s_m, for $m = 1, \ldots, \bar{m}$ to modify GNO (3) by,

$$\widetilde{w}_e^m(y) = \sum_{k=1}^{n_y^m} \alpha_k^m K^m(y, x_k, a(x_k))\varphi(a(x_k)), \tag{4}$$

for any fixed scale m and with MLPs K^m, φ. The above sum is taken over all the n_y^m points $x_k \in D_\Delta$ such that $|y - x_k| \leq r_m$. To choose the quadrature weights α_k^m, we propose an *attention based* choice,

$$\alpha_k^m = \frac{\exp(e_k^m)}{\sum_{k'=1}^{n_y^m} \exp(e_{k'}^m)}, \quad e_k^m = \frac{\langle \mathbf{W}_q^m y, \mathbf{W}_\kappa^m x_k \rangle}{\sqrt{\bar{d}}}, \tag{5}$$

with $\mathbf{W}_q^m, \mathbf{W}_\kappa^m \in \mathbb{R}^{\bar{d} \times m}$ are query and key matrices respectively, completing the description of the *attentional graph neural operator* (AGNO) (4) at each scale m.

Geometry Embeddings. The only geometric information in the afore-described encoder is provided by the coordinates of the underlying points. This alone does not convey the rich geometric information about the domain that can affect the solution of the underlying PDE (2). Hence, we need to embed further geometric information into the model. Deviating from the literature where geometric information is provided either by appending them as node and edge features on the underlying graphs [16] or by encoding a signed distance function [28], we propose to use novel *geometry embeddings* to encode this information. To this end and as described in **SM** Sec. B.3, we can rely on *local statistical embeddings* for each point $y \in \mathcal{D}$ as all the neighboring points x_k in D_Δ with $|y - x_k| \leq r_m$ have already been computed in the AGNO encoder. From these points, we can readily compute statistical

descriptors such as i) number of neighbors $x_k \in D_\Delta$, in the ball $B_{r_m}(y)$, ii) the *average distance* $D_{\text{avg}} = \frac{1}{n_y^m}\sum_{k=1}^{n_y^m}|y - x_k|$, iii) the variance of this distance D_{var}, with respect to the average D_{avg}, iv) the *centroid offset vector* $\Delta_y = \frac{1}{n_y^m}\sum_{k=1}^{n_x^y}(x_k - y)$ and v) a few principal component (PCA) features of the covariance matrix of $y - x_k$ to calculate the *local shape anisotropy*. These statistical descriptors, for each scale m and each point $y \in \mathcal{D}$ are then concatenated into a single vector z_y, normalized across components to yield zero mean and unit variance and fed into an MLP to provide the embedding $g^m(y)$. Alternatively, geometry embedding using *PointNet* models [42] can also be considered.

MAGNO. As shown in Fig. 2, the scale-dependent AGNO $\widetilde{w}_e^m$ (4) and the geometry embedding g^m, at each scale m, can be concatenated together and passed through another MLP to yield a scale-specific latent features function $\widehat{w}^m(y)$. Next, we need to integrate these features across all m scales. Instead of naively summing these scale contributions, we observe that different scales might contribute differently for every latent token to the encoding. To ascertain this relative contribution, we introduce a (small) MLP ψ_m and weigh the relative contributions with a *softmax* and combine them into the *multiscale attentional graph neural operator* or MAGNO encoder by setting,

$$w_e(y) = \sum_{m=1}^{\widetilde{m}} \beta_m(y)\widehat{w}^m(y), \quad \forall y \in \mathcal{D}, \quad \beta_m(y) = \frac{\exp(\psi_m(y))}{\sum_{m'=1}^{M}\exp(\psi_{m'}(y))} \tag{6}$$

Transformer Processor. The encoder provides a set of *geometry aware tokens* $w_e(y_\ell)$, for all points $y_\ell \in \mathcal{D}$, with $1 \le \ell \le L$, in the latent point cloud. These tokens are further transformed by a processor. As shown in Fig. 2, we choose a suitable transformer based processor. While postponing details on the processor architecture to **SM** Sec. B.4, we summarize our choices here. If the latent points $\{y_\ell\}$ lie on a regular grid (either through a structured stencil or a projected low-dimensional one), we use a patch-based *vision transformer* or ViT ([11] and [21, 38] for PDE operator learning) for computational efficiency. The equispaced latent points are combined into patches and the tokens in each patch are flattened into a single token embedding which serves as the input for a multi-head attention block, followed by a feed forward block. RMS normalization is applied to the tokens before processing. Either sinusoidal absolute position embeddings or rotary relative position embeddings are used to encode token positions. If the latent points y_ℓ are randomly downsampled from the original point cloud, there is no obvious way to patch them together. Hence, a standard transformer [50], but with RMS normalization, can be used. Additionally, we employ multiple skip connections across transformer blocks (see Fig. 2). The transformer processor transforms the tokens $w_e(y_\ell)$ into processed tokens, that we denote by $w_p(y_\ell)$, for all $1 \le \ell \le L$.

Decoder. Given any query point $x \in D$ in the original domain, the task of the decoder in GAOT is to provide $w(x)$, which approximates the solution u of the PDE (1) at that point. To this end, we simply employ the MAGNO architecture in reverse. By choosing a base radius $\hat{r}_0$ and scale factors $\hat{s}_m$, a set of increasing radii $\hat{r}_m = \hat{s}_m\hat{r}_0$ are selected to define a set of increasing balls $B_{\hat{r}_m}(x)$ around the query point x (Fig. 2). A corresponding AGNO model is defined by replacing $y \to x$, $x_k \to y_\ell$ and $a \to w_p$ in (4), with corresponding attentional weights computed via (5). In parallel, *geometry embeddings* over each ball $B_{\hat{r}_m}(x)$ are computed to provide statistical information about how the latent points y_ℓ are distributed in the neighborhood of the query point x. These AGNO features and geometry embeddings are concatenated and passed through a MLP to provide $w(x)$ which has the desired dimensions of the solution u of the PDE (1). We denote the GAOT model as $\mathbb{S}_\theta$ with the output $w = \mathbb{S}_\theta(a)$, for the inputs a to the PDE (1). It is trained to minimize the mismatch the underlying operator $\mathbb{S}$, i.e, the parameters θ are determined to minimize a loss $\mathcal{L}(\mathbb{S}(a), \mathbb{S}_\theta(a))$, over all input samples a^i, with $\mathcal{L}$ being either the absolute or mean-square errors.

Extension to time-dependent problems. To learn the solution operator $\mathbb{S}_t$ of the time-dependent PDE, we observe that the $\mathbb{S}_t$ can be used to update the solution forward in time, given the solution at any time point $u(t)$ by applying $u(t+\tau) = \mathbb{S}_\tau(u(t))$. Thus, for any time t, given the augmented input $a(t) = (c, u(t))$, with c being the coefficient in the PDE (2), we need GAOT to output $u(t+\tau)$, for any $\tau \ge 0$. To this end, we retain the architecture of GAOT, as described for the time-independent case above, and simply add the current time t and the *lead-time* τ as further inputs to the model.

Table 1: Benchmark results on time-dependent and time-independent datasets. Best and 2nd best models are shown in blue and orange fonts for each dataset.

Dataset	**Median relative L^1 error [%]**					
Time-Independent	**GAOT**	**RIGNO-18**	**Transolver**	**GNOT**	**UPT**	**GINO**
Poisson-C-Sines	**3.10**	**6.83**	77.3	100	100	20.0
Poisson-Gauss	**0.83**	2.26	**2.02**	88.9	48.4	7.57
Elasticity	**1.34**	**4.31**	4.92	10.4	12.6	4.38
NACA0012	**6.81**	**5.30**	8.69	6.89	16.1	9.01
NACA2412	**6.66**	**6.72**	8.51	8.82	17.9	9.39
RAE2822	6.61	**5.06**	**4.82**	7.15	16.1	8.61
Bluff-Body	**2.25**	5.76	**1.78**	44.2	5.81	3.49
Time-Dependent	**GAOT**	**RIGNO-18**	**GeoFNO**	**FNO DSE**	**UPT**	**GINO**
NS-Gauss	**2.91**	**2.29**	41.1	38.4	92.5	13.1
NS-PwC	**1.50**	**1.58**	26.0	56.7	100	5.85
NS-SL	**1.21**	**1.28**	13.7	22.6	51.5	4.48
NS-SVS	**0.46**	**0.56**	9.75	26.0	4.2	1.19
CE-Gauss	**6.40**	**6.90**	42.1	30.8	64.2	25.1
CE-RP	**5.97**	**3.98**	18.4	27.7	26.8	12.3
Wave-Layer	**5.78**	**6.77**	11.1	28.3	19.6	19.2
Wave-C-Sines	**4.65**	**5.35**	13.1	5.52	12.7	5.82

More precisely, the time-dependent version of GAOT is of the form $\widehat{\mathcal{S}}_\theta(x, t, \tau, a(t))$, where $a(t)$ takes values at points sampled in D. Following [39], the map $\hat{S}_\theta$ can be used to update an approximate solution of PDE (2) in time by following a very general time-stepping strategy:

$$\mathcal{S}_\theta(t, \tau, a(t)) = \gamma u(t) + \delta \widehat{\mathcal{S}}_\theta(x, t, \tau, a(t)). \tag{7}$$

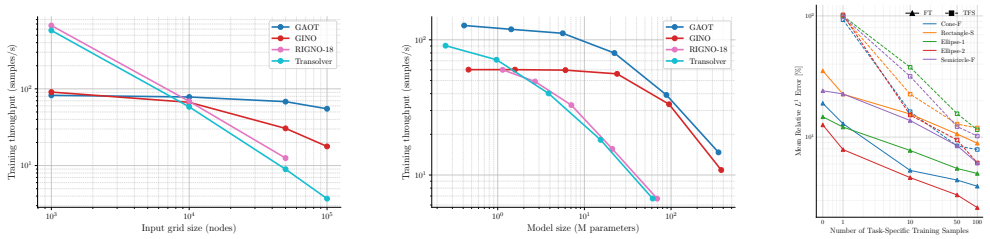
Here, choosing the parameters (γ, δ) appropriately leads to different strategies for time stepping: $\gamma = 0, \delta = 1$ directly approximates the *output* of the solution operator at time $t + \tau$; $\gamma = 1, \delta = 1$ yields the *residual* of the solution at the later time, with respect to the solution at current time; $\gamma = 1, \delta = \tau$ is equivalent to approximating the *time-derivative* of the solution. GAOT provides the flexibility to use any of these time-stepping strategies. We also use the *all2all* training strategy [21] to leverage trajectory data for time-dependent PDEs.

Efficient implementation. As our goal is to ensure accuracy and computational efficiency, we have designed GAOT with ability for large-scale computations in mind. We started with the realization that the heaviest burden of the computation should fall on the processor. The encoder and decoder are often responsible for memory overheads as these modules entail sparse computations on graphs with far more edges than nodes, making the computations largely edge-based and leading to high (and inefficient) memory usage. Moreover, in many PDE learning tasks on arbitrary geometries, the underlying domain (and the resulting graph) varies significantly between data samples, making load balancing very difficult.

To address these computational challenges, we resorted to i) moving the graph construction outside the model evaluation by either storing the graph, representing the input point cloud, in memory for small graphs or on disk for large graphs and loading them during training with efficient data loaders ii) sequentially processing each input in a given batch for the encoder and decoder, while still batch processing in the transformer processor, allowing us to reduce memory usage while retaining efficiency and iii) if needed for very large-scale datasets, we use an edge-dropping strategy to further increase the memory usage of the encoder and decoder. These innovations are essential to ensure batch training and underpin the efficiency of GAOT, even when input geometries vary significantly. A more detailed discussion on these *novel implementation tricks* is provided in SM E.2.

3 Results.

Datasets and Baselines. We start by testing GAOT of a challenging suite of 15 datasets for PDEs with input/output data on arbitrary point clouds in two space dimensions, see **SM** Secs D and G



(a) Grid vs. Throughput (b) Model vs. Throughput (c) Transfer Learning

Figure 3: Training throughput (samples/s) with increasing input grid size (a) and model size (b) for proposed GAOT, GINO, RIGNO and Transolver. (c) Transfer learning performance of GAOT on unseen bluff body shapes (See SM Sec. E.9 for dataset details). FT (fine-tuning) adapts a pretrained GAOT model from Table 1, while TFS denotes training from scratch. FT consistently outperforms TFS across varying numbers of task-specific training samples.

for a detailed description of the datasets and for visualizations, respectively. For time-independent PDEs, in addition to the elasticity benchmark of [25], we consider two Poisson equation datasets: Poisson-Gauss, defined on random points in a square domain, and Poisson-C-Sines, a new dataset we propose, containing rich multiscale solutions on a circular domain. In addition, we propose 4 new datasets comprising compressible flows past objects, both airfoils as well as bluff bodies. These datasets have significant variation in domain geometry and flow conditions (Mach numbers ranging from subsonic to supersonic, varying angles of attack etc.) and are tailor-made for testing neural PDE surrogates on arbitrary domains in two space dimensions. For time-dependent PDEs, we test on the challenging datasets considered recently in [39], composed of 8 operators corresponding to the compressible Euler (2), incompressible Navier-Stokes (4), and acoustic wave equations (2). These time-dependent operators include complex multiscale solutions with shocks and other sharp traveling waves which can interact, reflect and diffract making them hard to learn. We test GAOT on these datasets and compare them with several widely used neural operators for PDEs on arbitrary domains including those based on message passing (RIGNO [39]), Fourier Layers (GINO [28], GeoFNO [25], FNO DSE[30]) and Transformers (Transolver [52], UPT [1] and GNOT [19]).

Accuracy and Robustness. In Table 1, we present the relative test errors for the above datasets to observe that GAOT is very accurate on all of them, being either the best (10) or second-best (4) model on 14 of them. On average, over the time-independent datasets, GAOT is almost 50% more accurate than the second-best performing model (RIGNO-18) while on time-dependent datasets, it is slightly more accurate than the second-best performing model (RIGNO-18). What is even more noteworthy is the *robustness* of the performance of GAOT over all the datasets. As seen from Table 1, the accuracy of GAOT is uniformly good over all the datasets and does not deteriorate on any of them. On the other hand, all the baselines show significantly poor performance on outlier datasets. This robustness can be quantified in terms of a *robustness score* (see **SM** Sec. E.3) to find that GAOT is almost three times more robust on the time-independent datasets as the second-best model (RIGNO-18), while GAOT and RIGNO-18 are as robust as each other on the time-dependent datasets.

Computational Efficiency and Scalability. It is worth reiterating that the computational efficiency of an ML model is a significant marker of overall performance. We test efficiency in terms of two critical quantities, the *training throughput* and the *inference latency*. For a given input and model size, training throughput measures the number of samples that a model can process during training (forward pass, backward pass and gradient update) per unit time (in seconds) on a given compute system (GPU or CPU). The higher the training throughput, the faster the model can be trained. On the other hand, the inference latency is the amount of time it takes for a model to infer a single input. We present the throughput and latency for GAOT and three selected baselines (RIGNO-18 for Graph-based, GINO for FNO-based and Transolver for Transformer-based

Table 2: Comparison of model size (Params.), throughput (samples/s), and latency (ms) across GAOT and representative baselines.

Model	Params. (M)	Tput.	Latency
GAOT	5.62	97.5	6.966
GINO	6.04	60.4	8.455
RIGNO-18	2.69	50.3	12.74
Transolver	3.86	39.5	15.29

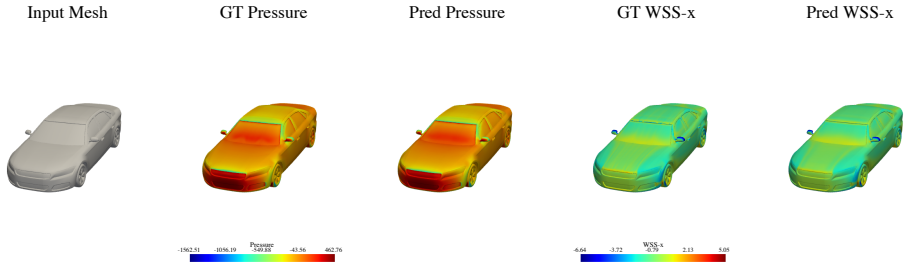


Figure 4: Comparison of predicted and ground-truth (GT) results for the pressure and wall shear stress in the x-direction (WSS-x) on the DrivAerNet++ test sample `N_S_WWS_WM_172`.

models) in Table 2 for learning tasks (such as the bluff-body dataset for compressible flow) where the domain geometry varies throughout the dataset. These experiments are conducted on one NVIDIA GeForce RTX 4090 GPU with float32 precision. We see from this table that GAOT has the highest training throughput and the fastest inference latency, being almost 50% and 15%, respectively better than the second-most efficient model (GINO).

How the training throughput of a model scales with increasing input and model size, is absolutely crucial for evaluating whether it can be used to process large-scale datasets (input scalability) or whether it can serve as a backbone of foundation models (model scalability) which require large model sizes [21]. To evaluate the scalability of different models, we plot how the training throughput changes as input size and model size (Fig. 3 (a, b)) are increased to find that GAOT scales much more favorably than the baselines with respect to both input and model size. In fact, models like Transolver and GNOT scale very poorly, making it impossible for us to train them for the large-scale time-dependent datasets with all2all training, which requires handling large volumes of data for large input sizes. Hence, we omit them in the accuracy results for time-dependent datasets in Tab. 1. The results for both accuracy and efficiency across a range of metrics for GAOT, RIGNO, GINO and Transolver are summarized in **SM** Tab. E.4 and visualized in the Radar chart Fig. 1. This demonstrates that GAOT ensures both accuracy (robustness) and computational efficiency (scalability) at the same time, while being the best model performing model on both sets of metrics.

Industrial scale 3D datasets. Given the high accuracy and excellent computational efficiency and scalability of GAOT, we showcase its abilities further on three challenging three-dimensional large-scale benchmarks for industrial simulations. We start with the DrivAerNet++ dataset of [13]. In this benchmark, the data consists of high-fidelity CFD simulations across 8K different car shapes which span the entire range of conventional car design. The underlying task is to learn steady-state surface fields (See Fig. 4) such as the pressure and wall shear stress, given the input car shape and flow conditions. The data has approximately 500K points per shape, making the overall training extremely compute intensive. Thus, only scalable models can currently process this learning task. We test GAOT on this challenging 3D benchmark and report the RMSE and MAE test errors for the pressure and wall shear stress in Tab. 3. Compared to baselines results taken from the leaderboard of the DrivAerNet++ challenge [9], we see that GAOT significantly improves on the state-of-the-art (see also Fig. 4). This improvement is most visible in the MAE for wall shear stress where GAOT is ca. 30% more accurate than the second-best model (TripNet), which currently sits atop the leaderboard for wall shear stress predictions. We recall that GAOT's decoder endows it with *neural field* properties. We showcase it for the DrivAerNet++ dataset by training GAOT on a randomly selected set of less than 10% of the total input points (per batch) and then testing on the original car surface point cloud by querying the desired points through GAOT's decoder. Although not as accurate as training GAOT with full input, we observe from Tab. 3 that this neural field version of GAOT has comparable accuracy to some of the baselines which have

Table 3: Error metrics of MSE ($\times 10^{-2}$) and Mean AE ($\times 10^{-1}$) for Pressure and Wall Shear Stress on the DrivAerNet++ dataset.

Model	**Pressure**		**Wall Shear Stress**	
	MSE	**Mean AE**	**MSE**	**Mean AE**
GAOT	4.2694	1.0699	8.6878	1.5429
FIGConvNet	4.9900	1.2200	9.8600	2.2200
TripNet	5.1400	1.2500	9.5200	2.1500
RegDGCNN	8.2900	1.6100	13.8200	3.6400
GAOT (NeurField)	12.0786	1.7826	22.9160	2.5099

Table 4: Comparison of GAOT and GINO across two benchmarks with MSE ($\times 10^{-2}$) and Mean AE ($\times 10^{-1}$). **Cp**: Pressure Coefficient, **WSS**: Wall Shear Stress, **P**: Pressure, **Cf**: Skin Friction Coefficient. **DML**: DrivaerML dataset, **CRM**: NASA CRM dataset.

Model	**Cp (DML)**		**WSS (DML)**		**P (CRM)**		**Cf (CRM)**	
	MSE	**Mean AE**	**MSE**	**Mean AE**	**MSE**	**Mean AE**	**MSE**	**Mean AE**
GAOT	5.1729	1.2352	16.9818	2.1640	7.7170	1.6014	16.1091	2.2305
GINO	8.8124	1.5238	28.4832	2.7330	10.5688	1.7450	21.1789	2.4240

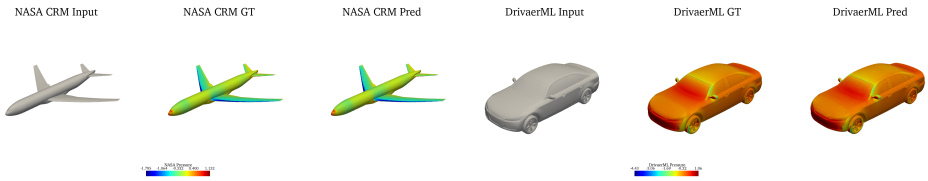


Figure 5: Comparison of predicted and ground-truth (GT) results for the pressure on the test sample of DrivAerML and NASA-CRM.

been trained with 10x more input points, further demonstrating the flexibility and accuracy of GAOT. Further assessments of the neural field property of GAOT are provided in SM E.8.

Next, we consider the very challenging DrivAerML dataset of [2] (see Fig. 5), where the learning task is exactly the same as in DrivAernet++, i.e., predicting surface fields on the car, given its shape as the input. However, unlike DrivAernet++ which was based on coarse RANS simulations, DrivAerML's ground truth is based on highly accurate LES simulations. This enables the incorporation of much more fine-scale physical effects in this dataset. The learning problem becomes harder, not just in terms of the challenging underlying physics, but also the fact that the number of points on the car surface is now 9M, instead of 500K for DrivAernet++. Thus, only highly scalable models can deal with this extreme resolution. Consequently, we are only able to test GAOT and GINO for this dataset and report the results in Table 4 to observe that GAOT is significantly (almost twice on wall shear-stress) as accurate as GINO on this dataset.

Finally, we consider an industrial-scale dataset, recently proposed in [6], where the learning task (see Fig. 5), is to predict surface pressure and the skin friction coefficient, given the shape of a full aircraft. The ground truth is generated with RANS simulations using a Spalart-Allmaras turbulence model, and the results with GAOT and GINO are reported in Table 4, showing that GAOT significantly outperforms GINO on this large-scale industrial dataset.

Generality, Generalization and Scaling. We highlight GAOT's flexibility with respect to the point distributions that it can handle by testing it on PDEs with regular grid inputs, as suggested in [39]. To this end, we considered 7 additional datasets and present the test errors in **SM** Sec. E.5. to find that GAOT is highly accurate even on regular grids and is either more accurate or comparable to the highly expressive GNN-based RIGNO, while being more accurate than widely used neural operators such as FNO and CNO. A key requirement in operator learning [23, 5] is the ability of the model to generalize (at test time) to input and output resolutions that are different from the training resolution. As GAOT can be readily evaluated at any query point, we showcase this aspect of GAOT in **SM** Sec. E.7. by plotting the test errors for a sequence of resolutions, different from the training resolution, for the Poisson-Gauss benchmark, to find that GAOT generalizes very well in both the sub- and super-resolution settings, even to grids with 10x more input points than the training resolution. Another test of the generalizability of a model is its ability to perform well *out-of-distribution*, either zero-shot or when it is fine-tuned with a few in-distribution samples for the new learning task. To test this aspect, we consider the datasets for compressible flow past bluff bodies and train a GAOT model on a set of bluff body shapes and then test it on shapes that were not in the training set. Then, the model is fine-tuned with a few task-specific samples and the results are shown in Fig. 3 (c). We observe that our model performs very well in a *few-shot transfer learning* scenario, with the fine-tuned model providing an almost order of magnitude gain in accuracy over the model, trained

from scratch. Finally, in **SM** Sec. E.6, we demonstrate that GAOT scales with both model and dataset size, with scaling with respect to dataset size, also illustrated in Fig. 3 (c)

Why does GAOT work so well ? To answer this question, we have performed extensive ablation studies in **SM** Sec.F to observe that i) the MAGNO encoder/decoder is clearly superior to message-passing based encoders/decoders, ii) choosing a regular equispaced latent point cloud performs significantly better than either downsampling on the original point cloud or using a projected low-dimensional regular grid, iii) GAOT is highly robust to the size of its latent grid, iv) a time-derivative marching strategy, i.e, setting $\gamma = 1, \delta = \tau$ in (7) is superior to other choices of γ, δ, v) using a statistical geometric embedding performs significantly better than either not using additional geometric information or using a pointnet to process geometric information vi) incorporating *multiscale* features in the MAGNO encoder/decoder provides a significant gain in accuracy when compared to using just a single scale GNO encoder/decoder as in GINO and vii) The power of GAOT does not just stem from its VIT processor, but also from its MAGNO encoder/decoder (SM E.5), acting in tandem. These results justify the choices that we have made in designing GAOT and selecting the relevant model components, while also revealing how these innovative features underpin GAOT's accuracy.

However, as argued before, this accuracy might come at the price of computational inefficiency. But, as we have demonstrated above, GAOT is also the most efficient model and does not have to pay the accuracy-efficiency trade-off. The reasons behind this boil down to the tricks used in efficiently implementing GAOT, which are discussed at length in SM E.2 and E.1.

4 Discussion

Summary. We present GAOT, a new neural operator for learning PDE solutions on arbitrary domains. It is based on a novel multiscale GNO encoder/decoder, combined with geometric embeddings that convey statistical information about the local domain geometry, and a (vision) transformer based processor. The model is designed to handle any point cloud input and provide the output at any query point in the underlying domain. Several innovative strategies have been used to make the implementation of GAOT computationally efficient and scalable. We test GAOT on a large number of challenging datasets for a variety of time-dependent and time-independent PDEs over diverse two-dimensional domain geometries to find that GAOT is significantly more accurate, robust and computationally efficient in terms of training throughput and inference latency, over a large set of baselines. We further demonstrate the potential of GAOT by presenting its SOTA performance on three large-scale three-dimensional datasets of industrial simulations in the automobile and aerospace sectors. These results demonstrate that GAOT can be a powerful and scalable neural operator with wide-spread applications. They also showcase the main advantage with an efficient and accurate neural operator such as GAOT, i.e, its inference time is many orders of magnitude faster than the runtime of a classical numerical PDE solver. We quantify this speedup in SM E.11 to find that GAOT can be anywhere between 4 to 9 orders of magnitude faster to run than classical PDE solvers, while retaining accuracy.

Related Work. As discussed before, there are 3 broad classes of models for learning PDEs on arbitrary domains namely i) end-to-end message-passing based frameworks exemplified here with RIGNO, which significantly improves upon models such as (multiscale) MeshGraphNets [41, 16] ii) Transformer based frameworks such as Transolver [52], GNOT [19] and UPT [1] and iii) frameworks, based on GNO encoders/decoders and FNO processors as in GINO [28]. GAOT differs from all these approaches by a) not using graph-based message passing, b) only employing transformers in the processor c) using a transformer, rather than FNO as a processor and significantly augmenting the GNO encoder/decoder by multiscale features, attention based-quadrature and geometry embeddings. It is precisely these choices, along with a highly efficient implementation, that allows GAOT to significantly surpass GINO, RIGNO, and Transolver in both accuracy and efficiency.

Limitations and Extensions. GAOT's excellent scalability and ability to generalize very well in a transfer learning setting (Fig. 3c) showcase its potential to serve as the backbone of foundation models for PDEs, extending models such as Poseidon [21] and DPOT [18] to arbitrary domains. Physics-informed loss functions can be added to GAOT to enable it to act as a physics-informed neural operator as in [29]. We also plan to apply GAOT to downstream tasks such as UQ [34], inverse problems, [37] and PDE constrained optimization [35] to further test its abilities. Finally, theoretical results for GAOT, such as universal approximation, will be considered in future work.