# Transolver: A Fast Transformer Solver for PDEs on General Geometries

**Haixu Wu** [1] **Huakun Luo** [1] **Haowen Wang** [1] **Jianmin Wang** [1] **Mingsheng Long** [1]

## Abstract

Transformers have empowered many milestones across various fields and have recently been applied to solve partial differential equations (PDEs). However, since PDEs are typically discretized into large-scale meshes with complex geometries, it is challenging for Transformers to capture intricate physical correlations directly from massive individual points. Going beyond superficial and unwieldy meshes, we present Transolver based on a more foundational idea, which is learning intrinsic physical states hidden behind discretized geometries. Specifically, we propose a new Physics-Attention to adaptively split the discretized domain into a series of learnable slices of flexible shapes, where mesh points under similar physical states will be ascribed to the same slice. By calculating attention to physics-aware tokens encoded from slices, Transovler can effectively capture intricate physical correlations under complex geometrics, which also empowers the solver with endogenetic geometry-general modeling capacity and can be efficiently computed in linear complexity. Transolver achieves consistent state-of-the-art with 22% relative gain across six standard benchmarks and also excels in large-scale industrial simulations, including car and airfoil designs. Code is available at https://github.com/thuml/Transolver.

## 1. Introduction

Solving partial differential equations (PDEs) is of immense importance in extensive real-world applications, such as weather forecasting, industrial design, and material analysis (Roubíček, 2013). As a basic scientific problem, it is usually hard to obtain analytic solutions for PDEs. Thus, PDEs are typically discretized into meshes and then solved by numerical methods in practice, which usually takes a few hours or even days for complex structures (Umetani & Bickel, 2018). Recently, deep models have emerged as promising tools for solving PDEs (Lu et al., 2021; Li et al., 2021). Benefiting from their impressive non-linear modeling capacity, they can learn to approximate the input and output mappings of PDE-governed tasks from data during training and then infer the solution significantly faster than numerical methods at the inference phase (Goswami et al., 2022; Wu et al., 2023).

As the major backbone of foundation models, Transformers (Vaswani et al., 2017) have achieved remarkable processes in extensive areas (Devlin et al., 2019; Brown et al., 2020; Dosovitskiy et al., 2021; Liu et al., 2021), which have also been introduced in PDE solving (Li et al., 2023c). However, as PDEs are usually discretized into large-scale meshes with complex geometrics for precise simulation, directly applying Transformers to massive mesh points faces difficulties in both computational efficiency and relation learning (Liu et al., 2021; Katharopoulos et al., 2020b), impeding them from being ideal PDE solvers. For instance, to calculate the drag force of a driving car (Figure 1), the model needs to approximate the solution of Navier-Stokes equations, including estimating the pressure for surface meshes and velocity for surrounding volumes, which poses the following two challenges. First, this problem involves collaborative modeling of tens of thousands of irregularly placed mesh points, which is computationally prohibited for canonical attention due to its quadratic complexity. Second, PDEs involve extremely complex spatiotemporal interactions among multiple physics quantities. It is hard to capture these high-order and intricate correlations directly from massive individual points. Thus, *how to efficiently capture physical correlations underlying the discretized domain* is the key to "transform" Transformers into practical PDE solvers.

Previous methods attempt to tackle the complexity problem by introducing linear attention (Hao et al., 2023; Tran et al., 2023), but directly applying the attention to massive mesh points may overwhelm the model from learning informative relations (Wu et al., 2022). In addition, solely relying on features of individual points is also insufficient in capturing intricate physical correlations of PDEs (Trockman & Kolter, 2022), especially for industrial design, which usually involves extremely complex multiphysics interactions. Besides, although the patchify operation is widely adopted to augment the feature of a single pixel with local informa-

[1]School of Software, BNRist, Tsinghua University. Haixu Wu <wuhx23@mails.tsinghua.edu.cn>. Correspondence to: Mingsheng Long <mingsheng@tsinghua.edu.cn>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

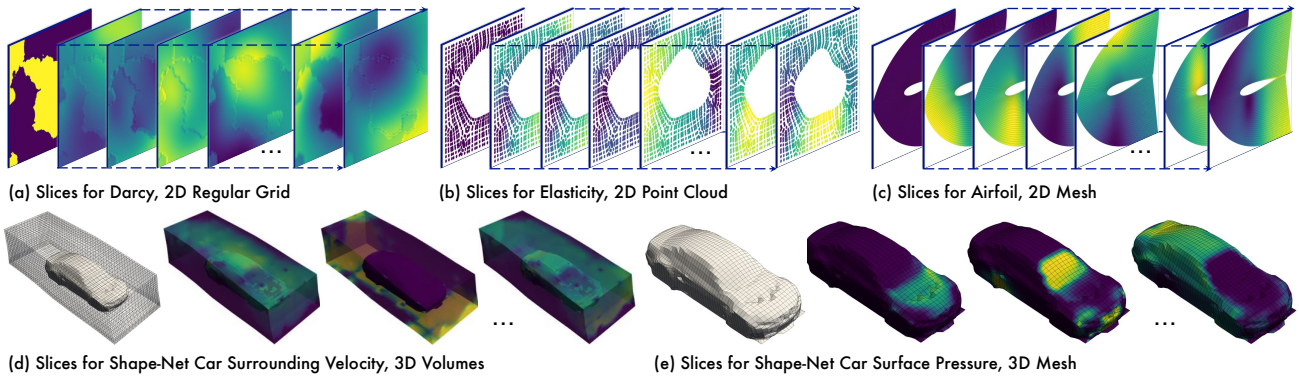



*Figure 1.* Visualization of learned slices in Transolver. For each case, the leftmost subfigure is model input and the right shows learned slices. A brighter color indicates the mesh point is more ascribed to the corresponding slice. See Appendix D for more visualizations.

tion in Vision Transformers (Dosovitskiy et al., 2021; Liu et al., 2021), the regular shape of patches is not applicable to unstructured geometries, let alone captures complicated physical states hidden under various discretized meshes.

Seeing through superficial and unwieldy meshes, this paper presents Transolver based on a more foundational idea, which is learning intrinsic physical states under complex geometrics. We present the Physics-Attention to decompose the discretized domain into a series of learnable slices, where mesh points under similar physical states will be ascribed to the same slice and then encoded into a physics-aware token. By applying attention to these learned physics-aware tokens, Physics-Attention can effectively capture complex underlying interactions behind the discretized domain. As shown in Figure 1, the learned slices clearly reflect miscellaneous *physical states* of PDEs, such as various fluid-structure interactions in a Darcy flow, different extrusion regions of elastic materials, shock wave and wake flow around the airfoil, front-back surfaces and up-bottom spaces of driving cars. This design can also natively adapt to intricate geometries and be effectively computed in linear time. We conduct extensive experiments on six well-established benchmarks with various geometries and large-scale industrial simulations, where Transolver achieves consistent state-of-the-art with impressive relative gain. Overall, our contributions are summarized as follows:

- Beyond prior methods, we propose to solve PDEs by learning intrinsic physical states behind the discretized domain, which frees our model from complex meshes and allows it to focus more on physical interactions.
- We present Transolver with Physics-Attention to decompose discretized domain into a series of learnable slices and apply attention to encoded physics-aware tokens, which can be computed in linear complexity.
- Transolver achieves consistent state-of-the-art with 22% relative gain across six standard benchmarks and excels in large-scale industrial simulations (e.g. car and airfoil designs), presenting favorable efficiency, scalability and out-of-distribution generalizability.

## 2. Related Work

### 2.1. Neural PDE Solvers

As a long-standing foundational problem in science and engineering, solving PDEs has gained significant attention. In the past centuries, various classical numerical methods, such as finite element method and spectral methods, have been proposed and widely used in practical applications (Wazwaz, 2002; Ŝolín, 2005). Recently, in view of the remarkable non-linear modeling capacity, deep models have also been introduced for solving PDEs as a fast surrogate (Karniadakis et al., 2021; Wang et al., 2023), which can be roughly categorized into the following two paradigms.

**Physics-informed neural networks** This paradigm formalizes PDE constraints, including equations, initial and boundary conditions as objective functions of deep models (Weinan & Yu, 2017; Raissi et al., 2019; Wang et al., 2020a;b). During training, the output of deep models will gradually satisfy PDE constraints, which can successfully approximate the PDE solution. However, this paradigm requires the exact formalization of PDEs, thus usually hard to apply to partially observed real-world applications.

**Neural operators** Another paradigm is to learn neural operators to approximate the input-output mappings in PDE-governed tasks, such as predicting the future fluid based on past observations or estimating the inner stress of solid materials (Lu et al., 2021; Kovachki et al., 2023). The most well-established models are FNO (Li et al., 2021) and its variants. Li et al. (2021) proposed Fourier neural operators by approximating integration with linear projection in the Fourier domain. Afterward, U-NO (Rahman et al., 2023) and U-FNO (Wen et al., 2022) are presented by plugging the FNO with U-Net (Ronneberger et al., 2015) for multiscale

modeling. Besides, WMT (Gupta et al., 2021) introduces multiscale wavelet bases to capture the complex correlations at various scales. F-FNO (Tran et al., 2023) enhances model efficiency by employing factorization on the Fourier domain. LSM (Wu et al., 2023) is recently proposed to tackle the high-dimension complexity of PDEs by applying spectral methods (Gottlieb & Orszag, 1977) in learned latent space.

To tackle irregular meshes, GNO (Li et al., 2020b) employs graph neural operators, and geo-FNO (Li et al., 2023b) utilizes the geometric Fourier transform to project the irregular input domain into uniform latent mesh. Recently, GINO (Li et al., 2023a) combines GNO and geo-FNO for local and global simultaneous modeling. 3D-GeoCA (Deng et al., 2024) enhances GNO by incorporating pre-trained 3D vision backbones (Xue et al., 2023) as better model initializations. However, due to the periodic boundary assumption of Fourier bases (Gottlieb & Orszag, 1977), geo-FNO will degenerate seriously in complex meshes, e.g. a car shape. Graph kernels also fall short in learning global information.

Especially, Transformers (Vaswani et al., 2017), as a vital cornerstone of deep learning, have also been applied to solve PDEs. HT-Net (Liu et al., 2022) integrates Swin Transformer (Liu et al., 2021) and multigrid method (Wesseling, 1995) to capture the multiscale spatial correlations. Fact-Former (Li et al., 2023d) utilizes the low-rank structure to boost the model efficiency with multidimensional factorized attention. However, these methods assume that PDEs are discretized into a uniform grid, limiting their applications in unstructured meshes. Besides, to address the quadratic complexity of attention, OFormer (Li et al., 2023c), GNOT (Hao et al., 2023) and ONO (Xiao et al., 2024) utilize the well-established linear Transformers, such as Reformer (Kitaev et al., 2020), Performer (Choromanski et al., 2021b) and Galerkin Transformer (Cao, 2021). Still, all of these methods directly apply attention to massive mesh points. In contrast, Transolver applies attention to intrinsic physical states captured by learnable slices, thereby better adept at modeling intricate physical correlations.

### 2.2. Geometric Deep Learning

A series of techniques have been developed to handle irregular geometrics, named geometric deep learning (Bronstein et al., 2017). Graph neural networks are representative ones, which employ kernels on connected graphs for representation learning (Hamilton et al., 2017; Gao & Ji, 2019; Pfaff et al., 2021). Besides, PointNet (Qi et al., 2017) and Point Transformer (Zhao et al., 2021) are also presented for scatter point clouds. However, most of these methods are proposed for computer vision or graphics, which are different from the PDE-solving task in this paper. Besides, all of these methods are well-designed for complex geometrics, while Transolver, benefiting from learning physical-sensitive slices, is apt at capturing physics information underlying unwieldy meshes.

## 3. Method

To tackle difficulties in efficiency and correlation modeling, we present Transolver with Physics-Attention to learn high-level correlations among intrinsic physical states under discretized meshes in PDE-governed tasks. Different from learning low-level relations over mesh points, focusing on physical states will free our model from complex geometrics, benefiting physics solving and computation efficiency.

**Problem setup** Consider PDEs defined on input domain $\Omega \subset \mathbb{R}^{C_{\mathbf{g}}}$, where $C_{\mathbf{g}}$ denotes the dimension of input space. For numerical calculation, $\Omega$ is firstly discretized into a finite set of $N$ mesh points $\mathbf{g} \in \mathbb{R}^{N \times C_{\mathbf{g}}}$. The task is to estimate target physical quantities based on input geometrics $\mathbf{g}$ and quantities $\mathbf{u} \in \mathbb{R}^{N \times C_{\mathbf{u}}}$ observed on $\mathbf{g}$. Here $\mathbf{u}$ is optional in some PDE-governed tasks. For instance, for fluid prediction, the input includes both the observation grid and observed past fluid velocity, where the target is the future velocity on each grid point. As for car or airfoil designs, the input only contains the discretized mesh structure and the model needs to estimate the surface and surrounding physics quantities.

### 3.1. Learning Physics-Aware Tokens

As we discussed before, the key to solving PDEs is to capture intricate physical correlations. However, the numerous discretized mesh points may overwhelm the attention mechanism from learning reliable correlations. Seeing through superficial meshes, we find that these mesh points are a finite discrete sampling of the underlying continuous physics space, which inspires us to learn the intrinsic physical states. As shown in Figure 2, which is to estimate the surface pressure of a driving car, we notice that the surface mesh set can be ascribed to several physically internal-consistent subsets, such as front, bevel and back areas. This discovery provides a more foundational view for solving PDE-governed tasks.

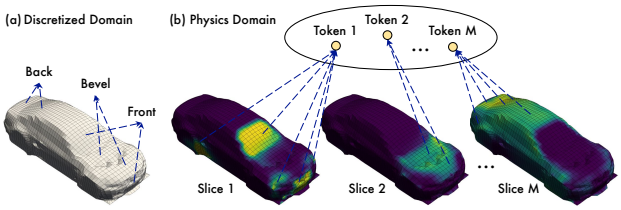



*Figure 2.* Learning physics-aware tokens from Transolver slices.

Technically, given a mesh set $\mathbf{g} = \{\mathbf{g}_i\}_{i=1}^N$ with the coordinate information of $N$ mesh points and observed quantities $\mathbf{u}$, we firstly embed them into deep features $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^N$ by a linear layer, where each mesh point feature contains $C$ channels, i.e. $\mathbf{x}_i \in \mathbb{R}^{1 \times C}$, and involves both geometry and physics information. To capture physical states under the whole input domain, we propose a bottom-up paradigm, that ascribes each mesh point $\mathbf{g}_i$ to $M$ potential slices based

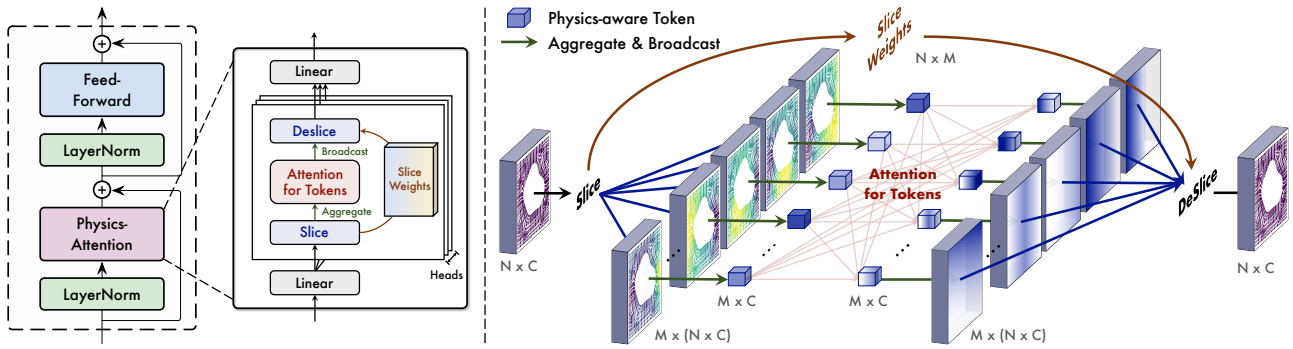



*Figure 3.* Overall design of Transolver layer, which replaces the standard attention with Physics-Attention. Each head encodes the input domain into a series of physics-aware tokens and then captures physical correlations under intricate geometrics by attention among tokens.

on its learned feature $\mathbf{x}_i$, which is formalized as follows:

$$\begin{aligned} \{\mathbf{w}_i\}_{i=1}^N &= \left\{\mathrm{Softmax}\left(\mathrm{Project}\left(\mathbf{x}_i\right)\right)\right\}_{i=1}^N \\ \mathbf{s}_j &= \{\mathbf{w}_{i,j}\mathbf{x}_i\}_{i=1}^N, \end{aligned} \tag{1}$$

where $\mathrm{Project}()$ projects $C$ channels into $M$ weights and yields slice weights $\mathbf{w}_i \in \mathbb{R}^{1\times M}$ after $\mathrm{Softmax}()$. Specifically, $\mathbf{w}_{i,j}$ represents the degree that the $i$-th mesh point belongs to the $j$-th slices with $\sum_{j=1}^M \mathbf{w}_{i,j} = 1$. $\mathbf{s}_j \in \mathbb{R}^{N\times C}$ represents the $j$-th slice feature, which is a weighted combination of $N$ mesh point features $\mathbf{x}$. Note that mesh points with close features will derive similar slice weights, which means they are more likely to be assigned to the same slice. To avoid a uniform assignment of each mesh point, we adopt $\mathrm{Softmax}()$ along the slice dimension (i.e. newly projected $M$ dimension) to make learned slice weights low-entropy and ensure informative physical states. In practice, $\mathrm{Project}()$ is configured as a point-wise linear layer, which can naturally adapt to general geometries. As for structured meshes or uniform grid, it can also be instantiated as a local convolution, mesh-free layer for better representations.

Afterward, since each slice contains mesh points with similar geometry and physics features, we further encode them into physical-aware tokens by spatially weighted aggregation, which can be written as follows:

$$\mathbf{z}_j = \frac{\sum_{i=1}^N \mathbf{s}_{j,i}}{\sum_{i=1}^N \mathbf{w}_{i,j}} = \frac{\sum_{i=1}^N \mathbf{w}_{i,j}\mathbf{x}_i}{\sum_{i=1}^N \mathbf{w}_{i,j}}, \tag{2}$$

where $\mathbf{z}_j \in \mathbb{R}^{1\times C}$. We normalize each token feature $\mathbf{z}_j$ by dividing the sum of slice weights. After encoding from physically internal-consistent slices by spatial aggregation, each token contains information of a specific physical state.

*Remark* 3.1 (**Why slices can learn physically internal-consistent information**). Firstly, as we aforementioned, slice weights are projected from mesh features. Thus, mesh points with similar features will be more likely to be assigned to the same slice. Secondly, since we will apply attention to the tokens encoded from slices, to decrease the final loss, the slice weights will be further optimized to assign mesh points under similar physical states to the same slice during training. Otherwise, the attention among tokens could be confused by the less distinguishable and state-hybrid token features, resulting in a less satisfying performance.

*Remark* 3.2 (**Learning slice is different from splitting computation area**). Classical numerical methods, such as finite element method, usually split the whole mesh into several computation areas for better simulation. This process requires huge specialized knowledge and manual effort (Ŝolín, 2005) and can only cover spatially local areas. It is insufficient to capture points under similar physical states but spatially distant, e.g. windshield and license plate of driving cars. In this paper, we take benefits from deep features and learn physical states in a bottom-up paradigm. The learned slices are beyond local areas. As shown in Figure 1(e), the model learns to ascribe the windshield, license plate and headlight of the car into the same slice because they are all in the front area during driving, which is highly related to the drag force, verifying the effectiveness of learning slices.

### 3.2. Transolver

Based on the idea of learning physics-aware tokens, we propose the Transolver by renovating Transformer with Physics-Attention to capture intricate physical correlations of PDEs.

**Physics-Attention** As described in the last section, for a deep feature $\mathbf{x} \in \mathbb{R}^{N\times C}$ embedded from input, we firstly decompose it into $M$ physically internal-consistent slices $\mathbf{s} = \{\mathbf{s}_j\}_{j=1}^M \in \mathbb{R}^{M\times(N\times C)}$ based on learned slice weights $\mathbf{w} \in \mathbb{R}^{N\times M}$. Then, to obtain the specific physics information contained in each slice, we aggregate $M$ slices to $M$ physics-aware tokens $\mathbf{z} = \{\mathbf{z}_j\}_{j=1}^M \in \mathbb{R}^{M\times C}$ by Eq. (2).

Next, as shown in Figure 3, we employ the attention mechanism among encoded tokens to capture intricate correlations among different physical states, that is

$$\mathbf{q}, \mathbf{k}, \mathbf{v} = \mathrm{Linear}(\mathbf{z}), \quad \mathbf{z}' = \mathrm{Softmax}\left(\frac{\mathbf{q}\mathbf{k}^{\mathsf{T}}}{\sqrt{C}}\right)\mathbf{v}, \tag{3}$$

where $\mathbf{q}, \mathbf{k}, \mathbf{v}, \mathbf{z}' \in \mathbb{R}^{M \times C}$. Afterward, transited physical tokens $\mathbf{z}' = \{\mathbf{z}'_j\}_{j=1}^{M}$ are transformed back to mesh points by deslicing, which recomposes tokens with slice weights:

$$\mathbf{x}'_i = \sum_{j=1}^{M} \mathbf{w}_{i,j} \mathbf{z}'_j, \tag{4}$$

where $1 \leq i \leq N$ and each token $\mathbf{z}'_j$ is broadcasted to all mesh points during above calculation. For clarity, we summarize the above process as $\mathbf{x}' = \text{Physics-Attn}(\mathbf{x})$, whose overall complexity is $\mathcal{O}(NMC + M^2C)$. Since we set $M$ as a constant and $M \ll N$, the computation complexity is linear w.r.t. the number of mesh points. Following the convention of attention mechanism (Vaswani et al., 2017), we adopt the multi-head version for Physics-Attention to augment the model capacity, which splits the input feature into several subspaces along the channel dimension.

*Remark* 3.3 (**Attention as learnable integral operator**). Prior methods define the PDE-solving task as an iterative updated process (Li et al., 2020b) and prove that canonical attention is a Monte-Carlo approximation of the integral operator on the input domain $\Omega$ (Cao, 2021; Kovachki et al., 2023), which can be used to approximate the solving process of each iteration step. However, in our work, the attention is applied to tokens encoded from slices. Toward a better theoretical understanding of Physics-Attention, we will prove that our design is also equivalent to learnable integral on $\Omega$.

**Theorem 3.4** (**Physics-Attention is equivalent to learnable integral on $\Omega$**). *Given input function $\boldsymbol{u} : \Omega \to \mathbb{R}^C$ and a mesh point $\mathbf{g}^* \in \Omega$, Physics-Attention is to approximate the integral operator $\mathcal{G}$, which is defined as:*

$$\mathcal{G}(\boldsymbol{u})(\mathbf{g}^*) = \int_{\Omega} \kappa(\mathbf{g}^*, \boldsymbol{\xi}) \boldsymbol{u}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}, \tag{5}$$

*where $\kappa(\cdot, \cdot)$ denotes the kernel function defined on $\Omega \times \Omega$.*

*Proof.* By constructing a diffeomorphism projection between mesh domain $\Omega$ and slice domain $\Omega_{\mathrm{s}}$, and substituting integration variable from $\boldsymbol{\xi} \in \Omega$ to $\boldsymbol{\xi}_{\mathrm{s}} \in \Omega_{\mathrm{s}}$, we can rewrite Eq. (5) as an integral on $\Omega_{\mathrm{s}}$, which is approximated by attention in Eq. (3). See Appendix A for complete proof. □

**Overall design** Following the architecture of canonical Transformer (Vaswani et al., 2017), we propose the Transolver by replacing the attention mechanism with Physics-Attention. Suppose there are $L$ layers, as shown in Figure 3, the $l$-th layer of Transolver can be formalized as follows:

$$\begin{aligned} \hat{\mathbf{x}}^l &= \text{Physics-Attn}\left(\text{LayerNorm}\left(\mathbf{x}^{l-1}\right)\right) + \mathbf{x}^{l-1} \\ \mathbf{x}^l &= \text{FeedForward}\left(\text{LayerNorm}\left(\hat{\mathbf{x}}^l\right)\right) + \hat{\mathbf{x}}^l, \end{aligned} \tag{6}$$

where $l \in \{1, \cdots, L\}$. $\mathbf{x}^l \in \mathbb{R}^{N \times C}$ is the output of the $l$-th layer. $\mathbf{x}^0 \in \mathbb{R}^{N \times C}$ represents the input deep feature, which is embedded from input geometries $\mathbf{g} \in \mathbb{R}^{N \times C_{\mathbf{g}}}$ and initial observation $\mathbf{u} \in \mathbb{R}^{N \times C_{\mathbf{u}}}$ by a linear embedding layer, i.e. $\mathbf{x}^0 = \text{Linear}\left(\text{Concat}(\mathbf{g}, \mathbf{u})\right)$. Here $C_{\mathbf{g}}$ is the dimension of geometry space and $C_{\mathbf{u}}$ is the number of observed physical quantities. At last, we adopt a linear projection upon $\mathbf{x}^L$ and obtain the final output as predictions of $\mathbf{u}$.

## 4. Experiments

We conduct extensive experiments to evaluate Transolver, including six well-established benchmarks and two industrial-level design tasks, covering various geometrics.

**Benchmarks** As presented in Table 1, our experiments span point cloud, structured mesh, regular grid and unstructured mesh in both 2D and 3D space. Elasticity, Plasticity, Airfoil, Pipe, Navier-Stokes and Darcy were proposed by FNO (Li et al., 2021) and geo-FNO (Li et al., 2022), which have been widely followed. Besides, we also experiment with car and airfoil design tasks. Shape-Net Car (Umetani & Bickel, 2018) is to estimate the surface pressure and surrounding air velocity given vehicle shapes. AirfRANS (Bonnet et al., 2022) contains high-fidelity simulation data for Reynolds-Averaged Navier–Stokes equations on airfoils from the National Advisory Committee for Aeronautics.

*Table 1.* Summary of experiment benchmarks, which includes various geometrics. #Mesh records the size of discretized meshes.

| Geometry | Benchmarks | #Dim | #Mesh |
|---|---|---|---|
| Point Cloud | Elasticity | 2D | 972 |
| Structured Mesh | Plasticity | 2D+Time | 3,131 |
| | Airfoil | 2D | 11,271 |
| | Pipe | 2D | 16,641 |
| Regular Grid | Navier–Stokes | 2D+Time | 4,096 |
| | Darcy | 2D | 7,225 |
| Unstructured Mesh | Shape-Net Car | 3D | 32,186 |
| | AirfRANS | 2D | 32,000 |

**Baselines** We comprehensively compare Transolver with more than 20 baselines, including typical neural operators: FNO (2021), U-NO (2023), LSM (2023), etc, Transformer PDE solvers: GNOT (2023), FactFormer (2023d), etc, and classical geometric deep models: PointNet (2017), GraphSAGE (2017), MeshGraphNet (2021), etc. LSM (Wu et al., 2023) and GNOT (Hao et al., 2023) are previous state-of-the-art on standard benchmarks. GINO (Li et al., 2023a) and 3D-GeoCA (Deng et al., 2024) are advanced deep models for large-scale industrial-level simulation benchmarks.

**Implementations** For fairness, we set the number of layers $L$ as 8 and the channel of hidden features $C$ as 128 or 256 according to the number of observed quantities of input data, which ensures that our model parameter is comparable to other Transformer-based models, such as GNOT

*Table 2.* Performance comparison on standard benchmarks. Relative L2 is recorded. A smaller value indicates better performance. For clarity, the best result is in bold and the second best is underlined. Promotion refers to the relative error reduction w.r.t. the second best model ($1 - \frac{\text{Our error}}{\text{The second best error}}$) on each benchmark. "/" means that the baseline cannot apply to this benchmark.

| Model | Point Cloud | Structured Mesh | | | Regular Grid | |
|---|---|---|---|---|---|---|
| | Elasticity | Plasticity | Airfoil | Pipe | Navier–Stokes | Darcy |
| FNO (Li et al., 2021) | / | / | / | / | 0.1556 | 0.0108 |
| WMT (Gupta et al., 2021) | 0.0359 | 0.0076 | 0.0075 | 0.0077 | 0.1541 | 0.0082 |
| U-FNO (Wen et al., 2022) | 0.0239 | 0.0039 | 0.0269 | 0.0056 | 0.2231 | 0.0183 |
| geo-FNO (Li et al., 2022) | 0.0229 | 0.0074 | 0.0138 | 0.0067 | 0.1556 | 0.0108 |
| U-NO (Rahman et al., 2023) | 0.0258 | 0.0034 | 0.0078 | 0.0100 | 0.1713 | 0.0113 |
| F-FNO (Tran et al., 2023) | 0.0263 | 0.0047 | 0.0078 | 0.0070 | 0.2322 | 0.0077 |
| LSM (Wu et al., 2023) | 0.0218 | 0.0025 | <u>0.0059</u> | 0.0050 | 0.1535 | <u>0.0065</u> |
| Galerkin (Cao, 2021) | 0.0240 | 0.0120 | 0.0118 | 0.0098 | 0.1401 | 0.0084 |
| HT-Net (Liu et al., 2022) | / | 0.0333 | 0.0065 | 0.0059 | 0.1847 | 0.0079 |
| OFormer (Li et al., 2023c) | 0.0183 | <u>0.0017</u> | 0.0183 | 0.0168 | 0.1705 | 0.0124 |
| GNOT (Hao et al., 2023) | <u>0.0086</u> | 0.0336 | 0.0076 | <u>0.0047</u> | 0.1380 | 0.0105 |
| FactFormer (Li et al., 2023d) | / | 0.0312 | 0.0071 | 0.0060 | 0.1214 | 0.0109 |
| ONO (Xiao et al., 2024) | 0.0118 | 0.0048 | 0.0061 | 0.0052 | <u>0.1195</u> | 0.0076 |
| **Transolver (Ours)** | **0.0064** | **0.0012** | **0.0053** | **0.0033** | **0.0900** | **0.0057** |
| Relative Promotion | 25.6% | 29.4% | 10.2% | 29.7% | 24.7% | 12.3% |

(2023) or ONO (2024). The number of slices $M$ is chosen from $\{32, 64\}$ according to the hidden dimension to balance model efficiency. All the experiments are conducted on one NVIDIA A100 GPU and repeated three times. In addition to the relative L2 of estimated physics fields, we also calculate the error and Spearman's rank correlation of drag and lift coefficients for practical design tasks. See Appendix B for comprehensive descriptions of implementations.

### 4.1. Main Results

**Standard Benchmarks** To clearly benchmark our model among multifarious neural operators, we first experiment on six well-established datasets, which can conveniently build a complete leaderboard from previous papers (Li et al., 2022; Wu et al., 2023; Hao et al., 2023).

As presented in Table 2, Transolver achieves consistent state-of-the-art across six widely-used benchmarks, covering solid and fluid physics in various geometrics. Notably, Transolver gains significant promotion in tasks on point cloud and structured mesh (25.6% in Elasticity, 29.4% in Plasticity, 29.7% in Pipe), demonstrating the effectiveness of our design in handling complex geometrics. Also, some advanced Transformer-based models, such as OFormer and GNOT, directly apply linear attention to mesh points, where we can find that they fall short in handling the Darcy benchmark. This is because Darcy requires the model to simulate the fluid pressure through the porous medium, which involves complex *fluid-structure interaction* (Bungartz & Schäfer, 2006) along the twisty medium boundary, while directly applying attention to massive mesh points will degenerate in correlation modeling (Wu et al., 2022). Benefiting from learning physical states, Transolver can obtain more informative geometry tokens than single mesh points and significantly reduce the token number, thereby capturing complex fluid-structure interactions better.

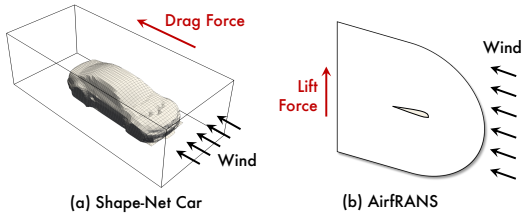



*Figure 4.* Car and airfoil design tasks. The key problem is to estimate the drag and lift force of a driving car or a flying airplane.

**Practical Design** Wind tunnel testing is one of the most essential steps in industrial design. As described in Figure 4, to examine the model effectiveness in complex real-world applications, we also experiment with the simulated wind tunnel scenario. Concretely, we simulate a driving car and record the surface pressure and surrounding air velocity, which can be used to calculate drag force. The task is to estimate the surface and surrounding physics fields based on the car's surface geometry. As for the AirfRANS, in addition to different airfoil shapes, our dataset also includes different angles of attack with different Reynolds numbers, which can better cover real flying cases. Note that these two tasks are quite challenging since they require the model to handle multiphysics simulation for hybrid geometrics.

As shown in Table 3, we can find that Transolver also excels in these two complex tasks among various geometric deep models and neural operators. In addition to achieving more accurate physics field estimation in both volume and surface,

*Table 3.* Performance comparison on practical design tasks. Both benchmarks are with unstructured mesh. In addition to the relative L2 of the surrounding (Volume) and surface (Surf) physics fields, the relative L2 of drag coefficient ($C_D$) and lift coefficient ($C_L$) is also recorded, along with their Spearman's rank correlations $\rho_D$ and $\rho_L$. A Spearman's correlation close to 1 indicates better performance.

| Model* | Shape-Net Car | | | | AirfRANS | | | |
|---|---|---|---|---|---|---|---|---|
| | Volume ↓ | Surf ↓ | $C_D$ ↓ | $\rho_D$ ↑ | Volume ↓ | Surf ↓ | $C_L$ ↓ | $\rho_L$ ↑ |
| Simple MLP | 0.0512 | 0.1304 | 0.0307 | 0.9496 | 0.0081 | 0.0200 | 0.2108 | 0.9932 |
| GraphSAGE (Hamilton et al., 2017) | 0.0461 | 0.1050 | 0.0270 | 0.9695 | 0.0087 | 0.0184 | <u>0.1476</u> | <u>0.9964</u> |
| PointNet (Qi et al., 2017) | 0.0494 | 0.1104 | 0.0298 | 0.9583 | 0.0253 | 0.0996 | 0.1973 | 0.9919 |
| Graph U-Net (Gao & Ji, 2019) | 0.0471 | 0.1102 | 0.0226 | 0.9725 | 0.0076 | 0.0144 | 0.1677 | 0.9949 |
| MeshGraphNet (Pfaff et al., 2021) | 0.0354 | 0.0781 | 0.0168 | 0.9840 | 0.0214 | 0.0387 | 0.2252 | 0.9945 |
| GNO (Li et al., 2020a) | 0.0383 | 0.0815 | 0.0172 | 0.9834 | 0.0269 | 0.0405 | 0.2016 | 0.9938 |
| Galerkin (Cao, 2021) | 0.0339 | 0.0878 | 0.0179 | 0.9764 | 0.0074 | 0.0159 | 0.2336 | 0.9951 |
| geo-FNO (Li et al., 2022) | 0.1670 | 0.2378 | 0.0664 | 0.8280 | 0.0361 | 0.0301 | 0.6161 | 0.9257 |
| GNOT (Hao et al., 2023) | 0.0329 | 0.0798 | 0.0178 | 0.9833 | <u>0.0049</u> | <u>0.0152</u> | 0.1992 | 0.9942 |
| GINO (Li et al., 2023a) | 0.0386 | 0.0810 | 0.0184 | 0.9826 | 0.0297 | 0.0482 | 0.1821 | 0.9958 |
| 3D-GeoCA (Deng et al., 2024) | <u>0.0319</u> | <u>0.0779</u> | <u>0.0159</u> | <u>0.9842</u> | / | / | / | / |
| **Transolver (Ours)** | **0.0207** | **0.0745** | **0.0103** | **0.9935** | **0.0037** | **0.0142** | **0.1030** | **0.9978** |

∗ Since both datasets are with unstructured mesh, not all the baselines are applicable. Concretely, the capability to handle unstructured mesh of typical neural operators (e.g., U-NO, LSM, etc) is based on geo-FNO (2022), which degenerates a lot in complex geometrics. Some Transformer-based models will also come across unstable training due to the massive mesh points, such as ONO and OFormer.

Transolver also performs best in design-oriented metrics, including drag and lift coefficient, as well as Spearman's rank correlation. Note that this metric measures the correlation between the ranking distribution of real coefficients and model-estimated values on all test samples, which quantifies the model's ability to rank different car or airfoil designs, therefore especially essential to shape optimization.

It is also worth noticing that geo-FNO degenerates seriously in Shape-Net Car. This is because geo-FNO transforms the input geometry into a uniform latent grid based on Fourier bases, which is insufficient for surface-volume hybrid geometry. Besides, 3D-GeoCA enhances GNO by employing the advanced 3D geometric deep model Point-BERT (Yu et al., 2022) as the feature encoder. However, even empowered with the advanced 3D geometric encoder, it still underperforms the Transolver. These results further demonstrate the advantages of our model in solving PDEs.

*Table 4.* Ablations on Physics-Attention. We experiment on two variants: changing the number of slices $M$ and replacing the learnable slices with fixed regular squares in $4 \times 4$ size (#Regular Squares). Efficiency is calculated on inputs with 1024 unstructured mesh points and batch size as 1. See Appendix C for full ablations.

| Ablations | | #Memory (GB) | #Time (s/epoch) | Relative L2 Elasticity | Relative L2 Darcy |
|---|---|---|---|---|---|
| Number of Slices | 1 | 0.60 | 37.76 | 0.0148 | 0.0386 |
| | 8 | 0.60 | 37.82 | 0.0071 | 0.0096 |
| | 16 | 0.61 | 37.96 | 0.0067 | 0.0067 |
| | 32 | 0.62 | 38.00 | 0.0067 | 0.0063 |
| | 64 | 0.64 | 38.18 | 0.0064 | 0.0059 |
| | 96 | 0.68 | 38.31 | 0.0061 | 0.0055 |
| | 128 | 0.69 | 38.78 | 0.0058 | 0.0054 |
| | 256 | 0.81 | 39.13 | **0.0054** | **0.0050** |
| | 512 | 1.01 | 39.75 | 0.0059 | 0.0056 |
| | 1024 | 1.53 | 40.49 | 0.0068 | 0.0055 |
| Regular Squares | | / | / | / | 0.0088 |

**Ablations** In addition to the main results, we also include ablations of our design in Table 4. In general, we can find that increasing the number of slices will benefit the final performance, which enables the model to capture more fine-grained physical states but also brings more computation costs. To balance efficiency and performance, we set $M$ as 64 for Elasticity and Darcy and choose it from $\{32, 64\}$ for other benchmarks. Note that when we set $M = 1$, the Physics-Attention will degenerate to a global pooling operator, which omits all the physical correlations and will cause a serious performance drop. Also, we can further observe that an extremely large $M$ (e.g. 1024) will slightly decrease the final performance. This may be because a too-large $M$ will make the physics domain seriously fragmented, resulting in too many tokens for subsequent attention calculation. In principle, the best choice of slice number is up to the physical property of the target PDE. In our experiments, $M$ is easy-to-tune in the range from 32 to 256.

Besides, replacing learnable slices with fixed regular squares will damage the model performance seriously, even for Darcy which is originally discretized into regular grids. This result further demonstrates the advantages of capturing physical states over solely computing in the discretized domain.

**Efficiency** To further demonstrate the model practicability, we provide the model efficiency comparison in Figure 6. We can find that in comparison with other Transformer-based models, Transolver presents favorable efficiency, consid-

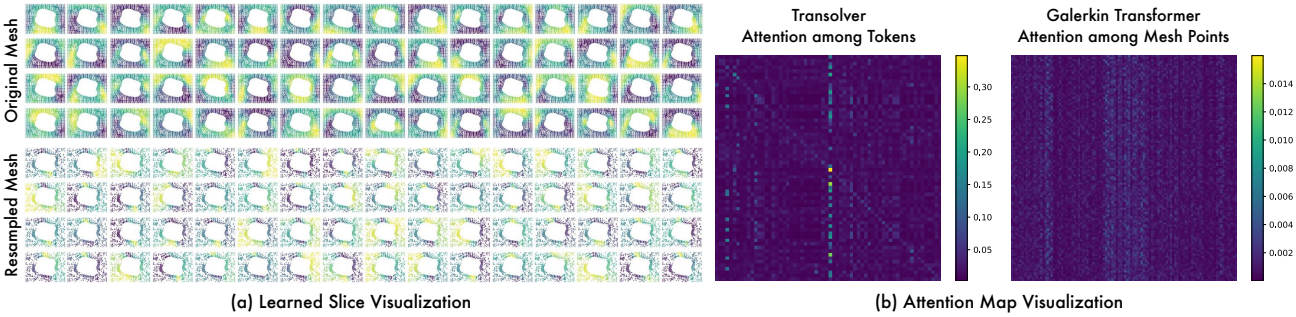



*Figure 5.* Physics-Attention visualization on Elasticity: (a) slice weights in the last layer of Transolver for both original and resampled meshes, (b) attention maps of the last layer in Transolver and Galerkin Transformer (Cao, 2021). See Appendix D.1 for more visualizations.

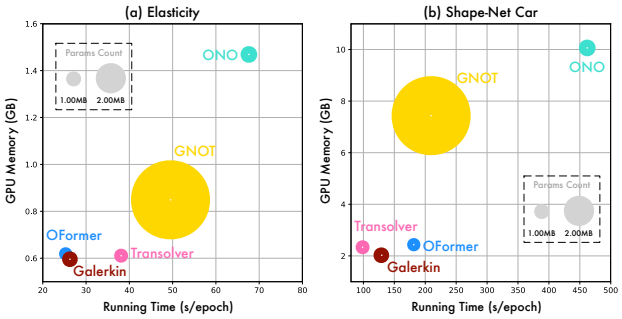



*Figure 6.* Efficiency comparison on Elasticity (972 mesh points) and Shape-Net Car (32,186 mesh points). Metrics are measured on the batch size as 1 and one epoch contains 1000 iterations. The growth curves w.r.t. input mesh size are provided in Appendix F.

ering running time, GPU memory and model parameters. Specifically, in Elasticity, Transolver achieves the 25.6% error reduction (0.0064 vs. 0.0086) with 5x fewer parameters and 1.3x running speed than the second-best model GNOT. Especially for large-scale meshes, our design of linear-complexity Physics-Attention benefits more significantly, where Transolver surpasses Galerkin Transformer and OFormer in both running time and final performance.

### 4.2. Model Analysis

**Physics-Attention analysis** Toward an intuitive understanding of Transolver, we visualize Physics-Attention in Figure 5. It is observed that slices can precisely capture mesh points under similar physics states. Also, it is worth noticing that we adopt the Softmax function in learning slice weights (Eq. (1)). Since Softmax function will make the learned weight distribution sharper, slices will also be optimized to capture more diverse patterns correspondingly, which empowers the model with better representation capability for intricate physics. For example, in the inner-stress estimation task of Elasticity, we can find that slices learn to cover diverse subareas that are under similar extrusion degrees, such as the left and right of the hollow area, corners of the material, etc. Further, learning physical states also frees our model from complex and unwieldy meshes.

*Table 5.* Comparison of linear attention in Galerkin Transformer and Physics-Attention in Transolver. Kullback–Leibler (KL) divergence between learned attention weights and a uniform distribution is recorded, which is averaged from the whole test set.

| BENCHMARKS | GALERKIN (CAO, 2021) | TRANSOLVER (OURS) |
|---|---|---|
| ELASTICITY (972 MESH POINTS) | 0.3803 | **1.7795** |
| DARCY (7,225 MESH POINTS) | 0.2739 | **1.8274** |

As shown in Figure 5(a), we randomly sample 50% input mesh of Elasticity. Surprisingly, Transolver can still capture physical states precisely even for broken meshes.

Besides, we also compare learned attention maps of the Transolver and Galerkin Transformer. As aforementioned, directly calculating attention among massive mesh points may overwhelm the model from learning informative relations (Wu et al., 2022). Statistical results in Table 5 present that the linear attention among mesh points is closer to a degenerated uniform distribution than Physics-Attention. As a supplement, we also include the attention visualization in Figure 5(b), where we can observe that the attention map among physics-aware tokens is much sharper than mesh-point attention. These results further verify the benefits of our physics-inspired design in facilitating attention learning.

**Case study** We plot error maps of different models in Figure 7 to provide a clear comparison. It is observed that Transolver performs significantly better in boundaries and multiple material junctions, such as the extrusion zone of elasticity, the medium boundary in Darcy and the curved region in the car surface. Note that these areas usually involve intricate physical interactions, which require the model to precisely capture the geometry information and its hidden physics, thereby further verifying the effectiveness of our design in handling complex geometrics. Especially, Transolver surpasses 3D-GeoCA (Deng et al., 2024) and GNOT (Hao et al., 2023) in predicting the wake flow behind the car and the pressure in the front area, demonstrating its capability in solving multiphysics PDEs on hybrid geometrics.

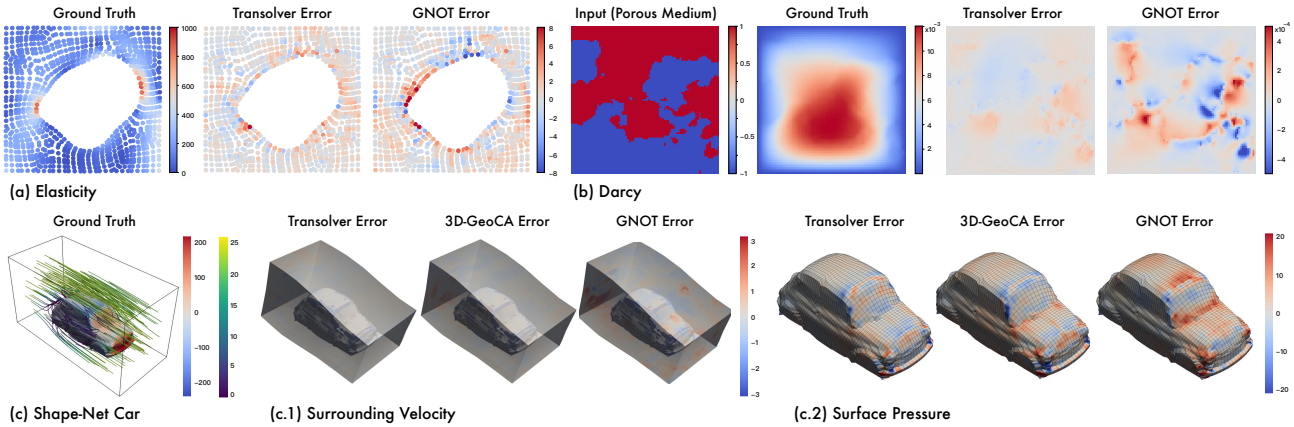



*Figure 7.* Case study on error maps of different models. Notably, Shape-Net Car requires to predict the surrounding velocity and surface pressure simultaneously. For clearness, we plot the error on volume and surface separately. See Appendix D.2 for more showcases.

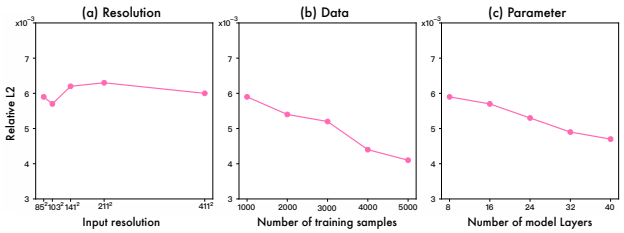



*Figure 8.* Model scalability on Darcy. We re-train the Transolver for PDEs on different (a) resolutions, (b) training samples and (c) model layers. See Appendix E.1 for full results on all benchmarks.

**PDE solving at scale** One well-acknowledged merit of Transformers is their scalability (Achiam et al., 2023). Thus, we also include a comprehensive test about the scalability of Transolver on resolution, data and model parameters in Figure 8. Specifically, we gradually increase the PDE resolution to 25x of the original setting, training data and model parameters to 5x. It is observed that Transolver can achieve consistent performance at scaled resolutions and benefit from more training data and larger model parameters, posing a potential for a large-scale pre-trained PDE solver.

**Out-of-distribution (OOD) generalization** In the previous research, neural solvers are mainly trained and tested with samples under the same or in-distribution PDE coefficients and varied initial or boundary conditions. For example, in the car design task, different samples of diverse shapes are all generated under the headwind with the same speed. As for the airfoil design, although training samples contain various Reynolds and angles of attacks, the test set is still under the same range of Reynolds and angles as the training set. Here, to further examine the generalizability of Transolver in real-world applications, we also experiment with OOD airfoil design tasks, where the test set has completely different Reynolds and angles of attacks.

As presented in Table 6, Transolver can handle OOD samples well, where it consistently performs best with Spearman's rank correlations of nearly 99% on unseen Reynolds and angles of attacks. These results indicate that Transolver not only fits the training data but also captures some generalizable physical information, further highlighting the advantage of calculating attention among physical states.

*Table 6.* OOD generalization experiments on the AirfRANS. Relative error of lift coefficient ($C_L$) and Spearman's rank correlations ($\rho_L$) are recorded. See Appendix E.3 for complete results.

| Models | OOD Reynolds | | OOD Angles | |
|---|---|---|---|---|
| | $C_L \downarrow$ | $\rho_L \uparrow$ | $C_L \downarrow$ | $\rho_L \uparrow$ |
| Simple MLP | 0.6205 | 0.9578 | 0.4128 | 0.9572 |
| GraphSAGE (2017) | 0.4333 | 0.9707 | <u>0.2538</u> | 0.9894 |
| PointNet (2017) | 0.3836 | 0.9806 | 0.4425 | 0.9784 |
| Graph U-Net (2019) | 0.4664 | 0.9645 | 0.3756 | 0.9816 |
| MeshGraphNet (2021) | 1.7718 | 0.7631 | 0.6525 | 0.8927 |
| GNO (2020a) | 0.4408 | <u>0.9878</u> | 0.3038 | 0.9884 |
| Galerkin (2021) | 0.4615 | 0.9826 | 0.3814 | 0.9821 |
| GNOT (2023) | <u>0.3268</u> | 0.9865 | 0.3497 | 0.9868 |
| GINO (2023a) | 0.4180 | 0.9645 | 0.2583 | <u>0.9923</u> |
| **Transolver (Ours)** | **0.2996** | **0.9896** | **0.1500** | **0.9950** |

## 5. Conclusions and Future Work

This paper presents Transolver to solve PDEs on general geometrics. Unlike prior Transformer operators, Transolver proposes to apply attention to learned physical states, which empowers our model with endogenetic geometry-general capacity and benefits physical correlations modeling. Transolver not only performs impressively on well-established benchmarks but also excels in practical design tasks, which consist of extremely complex geometrics and tanglesome multiphysics interactions. Extensive analyses are provided to verify the model's performance, efficiency, scalability and out-of-distribution generalizability. In the future, we will further explore the large-scale pre-training of Transolver in pursuit of foundation models for PDE solving.

## Acknowledgements

This work was supported by the National Key Research and Development Plan (2021YFC3000905), the National Natural Science Foundation of China (U2342217 and 62022050), the BNRist Innovation Fund (BNR2024RC01010), and the National Engineering Research Center for Big Data Software. We thank our colleagues Hang Zhou and Yuezhou Ma for their suggestions in the OOD experiments of this paper.

## Impact Statement

This paper presents work whose goal is to advance the deep learning research for solving PDE. The proposed method is based on a new idea of capturing correlations among learned physical states, which could be inspiring for future research. And our model also performs well in large-scale design tasks, which can be useful in industrial production. Note that this paper mainly focuses on the scientific problem. When developing our approach, we are fully committed to ensuring ethical considerations are taken into account. Thus we believe there are no potential ethical risks in our work.

## References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Bonnet, F., Mazari, J. A., Cinnella, P., and patrick gallinari. AirfRANS: High fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier–stokes solutions. In *NeurIPS Datasets and Benchmarks Track*, 2022.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In *NeurIPS*, 2020.

Bungartz, H.-J. and Schäfer, M. *Fluid-structure interaction: modelling, simulation, optimisation*. Springer Science & Business Media, 2006.

Cao, S. Choose a transformer: Fourier or galerkin. In *NeurIPS*, 2021.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Chen, T., Xu, B., Zhang, C., and Guestrin, C. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016.

Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlós, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L. J., and Weller, A. Rethinking attention with performers. *ICLR*, 2021a.

Choromanski, K. M., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J. Q., Mohiuddin, A., Kaiser, L., Belanger, D. B., Colwell, L. J., and Weller, A. Rethinking attention with performers. In *ICLR*, 2021b.

Deng, J., Li, X., Xiong, H., Hu, X., and Ma, J. Geometry-guided conditional adaption for surrogate models of large-scale 3d PDEs on arbitrary geometries. In *IJCAI*, 2024.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

Dym, C. L., Shames, I. H., et al. *Solid mechanics*. Springer, 1973.

Gao, H. and Ji, S. Graph u-nets. In *ICML*, 2019.

Goswami, S., Kontolati, K., Shields, M. D., and Karniadakis, G. E. Deep transfer operator learning for partial differential equations under conditional shift. *Nat. Mach. Intell*, 2022.

Gottlieb, D. and Orszag, S. A. *Numerical analysis of spectral methods: theory and applications*. SIAM, 1977.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. In *NeurIPS*, 2021.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. *NeurIPS*, 2017.

Hao, Z., Ying, C., Wang, Z., Su, H., Dong, Y., Liu, S., Cheng, Z., Zhu, J., and Song, J. Gnot: A general neural operator transformer for operator learning. *ICML*, 2023.

Hubbert, M. K. Darcy's law and the field equations of the flow of underground fluids. *Transactions of the AIME*, 1956.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nat. Rev. Phys.*, 2021.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020a.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *ICLR*, 2020.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *JMLR*, 2023.

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*, 2020a.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *ICLR*, 2021.

Li, Z., Kovachki, N. B., Choy, C., Li, B., Kossaifi, J., Otta, S. P., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., and Anandkumar, A. Geometry-informed neural operator for large-scale 3d PDEs. In *NeurIPS*, 2023a.

Li, Z., Kovachki, N. B., Choy, C., Li, B., Kossaifi, J., Otta, S. P., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., et al. Geometry-informed neural operator for large-scale 3d pdes. *arXiv preprint arXiv:2309.00583*, 2023b.

Li, Z., Meidani, K., and Farimani, A. B. Transformer for partial differential equations' operator learning. *TMLR*, 2023c.

Li, Z., Shu, D., and Farimani, A. B. Scalable transformer for pde surrogate modeling. *NeurIPS*, 2023d.

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

Liu, X., Xu, B., and Zhang, L. HT-net: Hierarchical transformer based operator learning model for multiscale PDEs. *arXiv preprint arXiv:2210.10890*, 2022.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. C.-F., and Guo, B. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, 2021.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR*, 2019.

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. *Nat. Mach. Intell*, 2021.

McCormick, B. W. *Aerodynamics, aeronautics, and flight mechanics*. John Wiley & Sons, 1994.

McLean, D. Continuum fluid mechanics and the navier-stokes equations. *Understanding Aerodynamics: Arguing from the Real Physics*, 2012.

Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., and Battaglia, P. Learning mesh-based simulation with graph networks. In *ICLR*, 2021.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

Rahman, M. A., Ross, Z. E., and Azizzadenesheli, K. U-no: U-shaped neural operators. *TMLR*, 2023.

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. *J. Comput. Phys.*, 2019.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

Roubíček, T. *Nonlinear partial differential equations with applications*. Springer Science & Business Media, 2013.

Sanchez-Gonzalez, A., Godwin, J., Pfaff, T., Ying, R., Leskovec, J., and Battaglia, P. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, 2020.

Ŝolín, P. *Partial differential equations and the finite element method*. John Wiley & Sons, 2005.

Spearman, C. The proof and measurement of association between two things. 1961.

Tran, A., Mathews, A., Xie, L., and Ong, C. S. Factorized fourier neural operators. In *ICLR*, 2023.

Trockman, A. and Kolter, J. Z. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

Umetani, N. and Bickel, B. Learning three-dimensional flow for interactive aerodynamic design. *ACM Transactions on Graphics (TOG)*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *NeurIPS*, 2017.

Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Van Katwyk, P., Deac, A., et al. Scientific discovery in the age of artificial intelligence. *Nature*, 2023.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient pathologies in physics-informed neural networks. *SIAM J. Sci. Comput.*, 2020a.

Wang, S., Yu, X., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *J. Comput. Phys.*, 2020b.

Wazwaz, A. M. Partial differential equations : methods and applications. 2002.

Weinan, E. and Yu, T. The deep ritz method: A deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.*, 2017.

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. U-fno–an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 2022.

Wesseling, P. Introduction to multigrid methods. Technical report, 1995.

Wu, H., Wu, J., Xu, J., Wang, J., and Long, M. Flowformer: Linearizing transformers with conservation flows. In *ICML*, 2022.

Wu, H., Hu, T., Luo, H., Wang, J., and Long, M. Solving high-dimensional pdes with latent spectral models. In *ICML*, 2023.

Xiao, Z., Hao, Z., Lin, B., Deng, Z., and Su, H. Improved operator learning by orthogonal attention. In *ICML*, 2024.

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G. M., Li, Y., and Singh, V. Nyströmformer: A nyström-based algorithm for approximating self-attention. *AAAI*, 2021.

Xue, L., Yu, N., Zhang, S., Li, J., Martín-Martín, R., Wu, J., Xiong, C., Xu, R., Niebles, J. C., and Savarese, S. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. *arXiv preprint arXiv:2305.08275*, 2023.

Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., and Lu, J. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *CVPR*, 2022.

Zhao, H., Jiang, L., Jia, J., Torr, P. H., and Koltun, V. Point transformer. In *ICCV*, 2021.

## A. Proof of Theorem 3.4

Firstly, we would like to present how to formalize canonical attention into a Monte-Carlo approximation of an integral operator as a Lemma, which is summarized from proofs provided by Cao (2021) and Kovachki et al. (2023).

**Lemma A.1.** *The canonical attention mechanism in Transformers is a Monte-Carlo approximation of an integral operator.*

*Proof.* Given input function $\boldsymbol{u}:\Omega\to\mathbb{R}^C$, the integral operation $\mathcal{G}$ defined on the function space $\Omega\to\mathbb{R}^C$ is formalized as:

$$\mathcal{G}(\boldsymbol{u})(\mathbf{g}^*)=\int_\Omega \kappa(\mathbf{g}^*,\boldsymbol{\xi})\boldsymbol{u}(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}, \tag{7}$$

where $\mathbf{g}^*\in\Omega\subset\mathbb{R}^{C_\mathbf{g}}$ and $\kappa(\cdot,\cdot)$ denotes the kernel function defined on $\Omega$. According to the formalization of attention, we propose to define the kernel function as follows:

$$\kappa(\mathbf{g}^*,\boldsymbol{\xi})=\left(\int_\Omega \exp\left(\left(\mathbf{W_q}\boldsymbol{u}(\boldsymbol{\xi}')\right)\left(\mathbf{W_k}\boldsymbol{u}(\boldsymbol{\xi})\right)^\mathsf{T}\right)\mathrm{d}\boldsymbol{\xi}'\right)^{-1}\exp\left(\left(\mathbf{W_q}\boldsymbol{u}(\mathbf{g}^*)\right)\left(\mathbf{W_k}\boldsymbol{u}(\boldsymbol{\xi})\right)^\mathsf{T}\right)\mathbf{W_v}, \tag{8}$$

where $\mathbf{W_q},\mathbf{W_k},\mathbf{W_v}\in\mathbb{R}^{C\times C}$.

Suppose that there are $N$ discretized mesh points $\{\mathbf{g}_1,\cdots,\mathbf{g}_N\}$, where $\mathbf{g}_i\in\Omega\subset\mathbb{R}^{C_\mathbf{g}}$. Approximating the inner-integral in Eq. (8) by Monte-Carlo, we have:

$$\int_\Omega \exp\left(\left(\mathbf{W_q}\boldsymbol{u}(\boldsymbol{\xi}')\right)\left(\mathbf{W_k}\boldsymbol{u}(\boldsymbol{\xi})\right)^\mathsf{T}\right)\mathrm{d}\boldsymbol{\xi}'\approx\frac{|\Omega|}{N}\sum_{i=1}^N\exp\left(\left(\mathbf{W_q}\boldsymbol{u}(\mathbf{g}_i)\right)\left(\mathbf{W_k}\boldsymbol{u}(\boldsymbol{\xi})\right)^\mathsf{T}\right). \tag{9}$$

Applying the above equation to Eq. (7) and using the same approximation for the outer-integral, we have:

$$\mathcal{G}(\boldsymbol{u})(\mathbf{g}^*)\approx\sum_{i=1}^N\frac{\exp\left(\left(\mathbf{W_q}\boldsymbol{u}(\mathbf{g}^*)\right)\left(\mathbf{W_k}\boldsymbol{u}(\mathbf{g}_i)\right)^\mathsf{T}\right)\mathbf{W_v}\boldsymbol{u}(\mathbf{g}_i)}{\sum_{j=1}^N\exp\left(\left(\mathbf{W_q}\boldsymbol{u}(\mathbf{g}_j)\right)\left(\mathbf{W_k}\boldsymbol{u}(\mathbf{g}_i)\right)^\mathsf{T}\right)}, \tag{10}$$

which is the calculation of the attention mechanism with $\mathbf{W_q},\mathbf{W_k},\mathbf{W_q}$ as linear layers for queries, keys and values. □

*Remark* A.2 (**Solving PDEs by learning integral neural operators**). Li et al. (2020a) firstly formalized the PDE-solving task as learning neural operators and defined the operator learning process as an iterative architecture, which corresponds to multiple layers in deep models. Then, Li et al. (2021) further defined each iteration step as a composition of a non-local integral operator and a local, nonlinear activation function. Since the nonlinear activation function can be easily parameterized by the feedforward layer. The key to solving PDEs is learning non-local integral operators. Thus, Lemma A.1 proves that a Transformer model can theoretically work as a neural operator and be applied to solve PDEs.

**Lemma A.3.** *Suppose that $\Omega$ is a countable domain, the slice domain $\Omega_\mathrm{s}$ is isomorphic to $\Omega$.*

*Proof.* For the $i$-th element $\mathbf{g}_i\in\Omega$, we define its slice weight to the $j$-th slice $\mathbf{s}_j$ as $w_{\mathbf{g}_i,\mathbf{s}_j}\in\mathbb{R}$. Given a constant $K\geq 1$ and $K\in\mathbb{N}$, since input domain $\Omega$ is countable, we can construct the projection $\boldsymbol{g}$ between input domain and slice domain $\Omega_\mathrm{s}$ as follows: for $i$ iterated from 1 to $+\infty$, its projected slice is defined as

$$\boldsymbol{g}(\mathbf{g}_i)=\underset{\mathbf{s}_j,\text{ where }\left(\lfloor\frac{i-1}{K}\rfloor\times K\right)<j\leq\left(\left(\lfloor\frac{i-1}{K}\rfloor+1\right)\times K\right)}{\arg\max}w_{\mathbf{g}_i,\mathbf{s}_j}\ \text{ subject to } j\text{-th slice is not projected before.} \tag{11}$$

Here we construct a bijection between the input domain and the slice domain. Thus, $\Omega\cong\Omega_\mathrm{s}$. □

Next, we will prove Theorem 3.4, which provides a theoretical understanding of our design in Physics-Attention. The formalization of Physics-Attention can be directly derived from integral based on proper assumptions.

*Proof.* According to Lemma A.3, we can obtain an isomorphic projection $\boldsymbol{g}$ between a countable input domain $\Omega$ and slice domain $\Omega_\mathrm{s}$. Suppose that the slice weight $w_{*,*}:\overline{\Omega}\times\overline{\Omega}_\mathrm{s}\to\mathbb{R}$ is smooth in both $\overline{\Omega}$ and $\overline{\Omega}_\mathrm{s}$, where the $\overline{\Omega}$ and $\overline{\Omega}_\mathrm{s}$ denote the continuation of $\Omega$ and $\Omega_\mathrm{s}$ respectively, we can obtain $\boldsymbol{g}$ as a diffeomorphism projection.

Then, we define the value function $\boldsymbol{u}_{\mathrm{s}}$ on the physics-aware token domain $\Omega_{\mathrm{s}}$ as follows:

$$\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s}}) = \left(\int_{\Omega} w_{\boldsymbol{\xi},\boldsymbol{\xi}_{\mathrm{s}}} \boldsymbol{u}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}\right) / \left(\int_{\Omega} w_{\boldsymbol{\xi},\boldsymbol{\xi}_{\mathrm{s}}} \mathrm{d}\boldsymbol{\xi}\right), \tag{12}$$

which corresponds to the slice token definition in Eq. (2).

Based on the above assumptions and definitions, we have:

$$\begin{aligned}
\mathcal{G}(\boldsymbol{u})(\mathbf{g}) &= \int_{\Omega} \kappa(\mathbf{g},\boldsymbol{\xi})\boldsymbol{u}(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi} \\
&= \int_{\Omega_{\mathrm{s}}} \kappa_{\mathrm{ms}}\big(\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}\big)\boldsymbol{u}_{\mathrm{s}}\left(\boldsymbol{\xi}_{\mathrm{s}}\right)\mathrm{d}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}) && (\kappa_{\mathrm{ms}}(\cdot,\cdot):\Omega\times\Omega_{\mathrm{s}}\to\mathbb{R}^{C\times C} \text{ is a kernel function}) \\
&= \int_{\Omega_{\mathrm{s}}} \kappa_{\mathrm{ms}}\big(\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}\big)\boldsymbol{u}_{\mathrm{s}}\left(\boldsymbol{\xi}_{\mathrm{s}}\right)|\det(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}))|\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}} \\
&= \int_{\Omega_{\mathrm{s}}} \left(\frac{\int_{\Omega_{\mathrm{s}}} w_{\mathbf{g},\boldsymbol{\xi}'_{\mathrm{s}}}\kappa_{\mathrm{ss}}\big(\boldsymbol{\xi}'_{\mathrm{s}},\boldsymbol{\xi}_{\mathrm{s}}\big)\mathrm{d}\boldsymbol{\xi}'_{\mathrm{s}}}{\int_{\Omega_{\mathrm{s}}} w_{\mathbf{g},\boldsymbol{\xi}'_{\mathrm{s}}}\mathrm{d}\boldsymbol{\xi}'_{\mathrm{s}}}\right)\boldsymbol{u}_{\mathrm{s}}\left(\boldsymbol{\xi}_{\mathrm{s}}\right)|\det(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}))|\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}} && (\kappa_{\mathrm{ms}} \text{ is a linear combination of } \kappa_{\mathrm{ss}} \text{ with weights } w_{*,*}) \\
&= \int_{\Omega_{\mathrm{s}}} \underbrace{w_{\mathbf{g},\boldsymbol{\xi}'_{\mathrm{s}}}}_{\text{DeSlice}} \int_{\Omega_{\mathrm{s}}} \underbrace{\kappa_{\mathrm{ss}}\big(\boldsymbol{\xi}'_{\mathrm{s}},\boldsymbol{\xi}_{\mathrm{s}}\big)}_{\text{Attention among slice tokens}} \underbrace{\boldsymbol{u}_{\mathrm{s}}\left(\boldsymbol{\xi}_{\mathrm{s}}\right)}_{\text{Slice token}}|\det(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}))|\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}\mathrm{d}\boldsymbol{\xi}'_{\mathrm{s}} && \left(\text{Suppose that } \int_{\Omega_{\mathrm{s}}} w_{\mathbf{g},\boldsymbol{\xi}'_{\mathrm{s}}}\mathrm{d}\boldsymbol{\xi}'_{\mathrm{s}} = 1\right) \\
&\approx \underbrace{\sum_{j=1}^{M}\mathbf{w}_{i,j}}_{\text{Eq. (4)}} \underbrace{\sum_{t=1}^{M}\frac{\exp\left(\left(\mathbf{W}_{\mathbf{q}}\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},j})\right)\left(\mathbf{W}_{\mathbf{k}}\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},t})\right)^{\mathsf{T}}/\tau\right)}{\sum_{p=1}^{M}\exp\left(\left(\mathbf{W}_{\mathbf{q}}\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},j})\right)\left(\mathbf{W}_{\mathbf{k}}\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},p})\right)^{\mathsf{T}}/\tau\right)}\mathbf{W}_{\mathbf{v}}}_{\text{Eq. (3)}} \underbrace{\left(\frac{\sum_{p=1}^{N}\mathbf{w}_{p,t}\boldsymbol{u}(\mathbf{g}_p)}{\sum_{p=1}^{N}\mathbf{w}_{p,t}}\right)}_{\text{Eq. (2)}} && \text{(Lemma A.1)} \\
&= \sum_{j=1}^{M}\mathbf{w}_{i,j}\sum_{t=1}^{M}\frac{\exp(\mathbf{q}_j\mathbf{k}_t^{\mathsf{T}}/\tau)}{\sum_{p=1}^{M}\exp(\mathbf{q}_j\mathbf{k}_p^{\mathsf{T}}/\tau)}\mathbf{v}_t,
\end{aligned} \tag{13}$$

where $\kappa_{\mathrm{ms}}$ defines the kernel function between mesh points and slices, and $\kappa_{\mathrm{ss}}$ is defined among slices. Since slices are permutation-invariant in our implementation, we take $|\det(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}))| = 1$ for simplification. Different from the attention among mesh points, the usage of Lemma A.1 here is based on the Monte-Carlo approximation in the slice domain. □

## B. Implementation Details

In this section, we provide the details of our experiments, including **benchmarks**, **metrics**, and **implementations**.

*Table 7.* Summary of experiment benchmarks, where the first six datasets are from FNO (Li et al., 2021) and geo-FNO (Li et al., 2022), Shape-Net Car is from (Umetani & Bickel, 2018) and preprocessed by (Deng et al., 2024), and AirfRANS is from (Bonnet et al., 2022). #Mesh records the size of discretized meshes. #Dataset is organized as the number of samples in training and test sets.

| GEOMETRY | BENCHMARKS | #DIM | #MESH | #INPUT | #OUTPUT | #DATASET |
|---|---|---|---|---|---|---|
| POINT CLOUD | ELASTICITY | 2D | 972 | STRUCTURE | INNER STRESS | (1000, 200) |
| STRUCTURED MESH | PLASTICITY | 2D+TIME | 3,131 | EXTERNAL FORCE | MESH DISPLACEMENT | (900, 80) |
| | AIRFOIL | 2D | 11,271 | STRUCTURE | MACH NUMBER | (1000, 200) |
| | PIPE | 2D | 16,641 | STRUCTURE | FLUID VELOCITY | (1000, 200) |
| REGULAR GRID | NAVIER–STOKES | 2D+TIME | 4,096 | PAST VELOCITY | FUTURE VELOCITY | (1000, 200) |
| | DARCY | 2D | 7,225 | POROUS MEDIUM | FLUID PRESSURE | (1000, 200) |
| UNSTRUCTURED MESH | SHAPE-NET CAR | 3D | 32,186 | STRUCTURE | VELOCITY & PRESSURE | (789, 100) |
| | AIRFRANS | 2D | 32,000 | STRUCTURE | VELOCITY & PRESSURE | (800, 200) |

### B.1. Benchmarks

We extensively evaluate our model in eight benchmarks, whose information is summarized in Table 7. Note that these benchmarks involve the following three types of PDEs:

- **Solid material** (Dym et al., 1973): Elasticity and Plasticity.
- **Navier-Stokes equations for fluid** (McLean, 2012): Airfoil, Pipe, Navier-Stokes, Shape-Net Car and AirfRANS.
- **Darcy's law** (Hubbert, 1956): Darcy.

Here are the details of each benchmark.

**Elasticity** This benchmark is to estimate the inner stress of the elasticity material based on the material structure, which is discretized in 972 points (Li et al., 2022). For each case, the input is a tensor in the shape of $972 \times 2$, which contains the 2D position of each discretized point. The output is the stress of each point, thus in the shape of $972 \times 1$. As for the experiment, 1000 samples with different structures are generated for training and another 200 samples are used for test.

**Plasticity** This benchmark is to predict the future deformation of the plasticity material under the impact from above by an arbitrary-shaped die (Li et al., 2022). For each case, the input is the shape of the die, which is discretized into the structured mesh and recorded as a tensor with shape $101 \times 31$. The output is the deformation of each mesh point in the future 20 time steps, that is a tensor in the shape of $20 \times 101 \times 31 \times 4$, which contains the deformation in four directions. Experimentally, 900 samples with different die shapes are used for model training and 80 new samples are for test.

**Airfoil** This task is to estimate the Mach number based on the airfoil shape, where the input shape is discretized into structured mesh with shape $221 \times 51$ and the output is the Mach number for each mesh point (Li et al., 2022). Here, all the shapes are deformed from the NACA-0012 case provided by the National Advisory Committee for Aeronautics. 1000 samples in different airfoil designs are used for training and the other 200 samples are for testing.

**Pipe** This benchmark is to estimate the horizontal fluid velocity based on the pipe structure (Li et al., 2022). Each case discretizes the pipe into structured mesh with size $129 \times 129$. Thus, for each case, the input tensor is in the shape of $129 \times 129 \times 2$, which contains the position of each discretized mesh point. The output is the velocity value for each point, thus in the shape of $129 \times 129 \times 1$. 1000 samples with different pipe shapes are used for model training and 200 new samples are for test, which are generated by controlling the centerline of the pipe.

**Navier-Stokes** This benchmark is to model the incompressible and viscous flow on a unit torus, where the fluid density is constant and viscosity is set as $10^{-5}$ (Li et al., 2021). The fluid field is discretized into $64 \times 64$ regular grid. The task is to predict the fluid in the next 10 steps based on the observations in the past 10 steps. 1000 fluids with different initial conditions are generated for training, and 200 new samples are used for test.

**Darcy** This benchmark is to model the flow of fluid through a porous medium (Li et al., 2021). Experimentally, the process is discretized into a $421 \times 421$ regular grid. Then we downsample the data into $85 \times 85$ resolution for main experiments. The input of the model is the structure of the porous medium and the output is the fluid pressure for each grid. 1000 samples are used for training and 200 samples are generated for test, where different cases contain different medium structures.

**Shape-Net Car** This benchmark focuses on the drag coefficient estimation for the driving car, which is essential for car design. Overall, 889 samples with different car shapes are generated to simulate the 72 km/h speed driving situation (Umetani & Bickel, 2018), where the car shapes are from the "car" category of ShapeNet (Chang et al., 2015). Concretely, they discretize the whole space into unstructured mesh with 32,186 mesh points and record the air around the car and the pressure over the surface. Here we follow the experiment setting in 3D-GeoCA (Deng et al., 2024), which takes 789 samples for training and the other 100 samples for testing. The input mesh of each sample is also preprocessed into the combination of mesh point position, signed distance function and normal vector. The model is trained to predict the velocity and pressure value for each point. Afterward, we can calculate the drag coefficient based on these estimated physics fields.

**AirfRANS** This dataset contains the high-fidelity simulation data for Reynolds-Averaged Navier–Stokes equations (Bonnet et al., 2022), which is also used to assist airfoil design. Different from Airfoil (Li et al., 2022), this benchmark involves more diverse airfoil shapes under finer discretized meshes. Specifically, it adopts airfoils in the 4 and 5 digits series of the National Advisory Committee for Aeronautics, which have been widely used historically. Each case is discretized into 32,000 mesh

points. By changing the airfoil shape, Reynolds number, and angle of attack, AirfRANS provides 1000 samples, where 800 samples are used for training and 200 for the test set. Air velocity, pressure and viscosity are recorded for surrounding space and pressure is recorded for the surface. Note that both drag and lift coefficients can be calculated based on these physics quantities. However, as their original paper stated, air velocity is hard to estimate for airplanes, making all the deep models fail in drag coefficient estimation (Bonnet et al., 2022). Thus, in the main text, we focus on the lift coefficient estimation and the pressure quantity on the volume and surface, which is essential to the take-off and landing stages of airplanes.

### B.2. Metrics

Since our experiment consists of standard benchmarks and practical design tasks, we also include several design-oriented metrics in addition to the relative L2 for physics fields.

**Relative L2 for physics fields** Given the physics field $\mathbf{u}$ and the model predicted field $\widehat{\mathbf{u}}$, the relative L2 of model prediction can be calculated as follows:

$$\text{Relative L2} = \frac{\|\mathbf{u} - \widehat{\mathbf{u}}\|}{\|\mathbf{u}\|}. \tag{14}$$

**Relative L2 for drag and lift coefficients** For Shape-Net Car and AirfRANS, we also calculated the drag and lift coefficients based on the estimated physics fields. For unit density fluid, the coefficient (drag or lift) is defined as follows:

$$C = \frac{2}{v^2 A}\left(\int_{\partial\Omega} p(\boldsymbol{\xi})\left(\widehat{n}(\boldsymbol{\xi})\cdot\widehat{i}(\boldsymbol{\xi})\right)\mathrm{d}\boldsymbol{\xi} + \int_{\partial\Omega}\tau(\boldsymbol{\xi})\cdot\widehat{i}(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}\right), \tag{15}$$

where $v$ is the speed of the inlet flow, $A$ is the reference area, $\partial\Omega$ is the object surface, $p$ denotes the pressure function, $\widehat{n}$ means the outward unit normal vector of the surface, $\widehat{i}$ is the direction of the inlet flow and $\tau$ denotes wall shear stress on the surface. $\tau$ can be calculated from the air velocity near the surface (McCormick, 1994), which is usually much smaller than the pressure item. Specifically, for the drag coefficient of Shape-Net Car, $\widehat{i}$ is set as $(-1, 0, 0)$ and $A$ is the area of the smallest rectangle enclosing the front of cars. As for the lift coefficient of AirfRANS, $\widehat{i}$ is set as $(0, 0, -1)$. The relative L2 is defined between the ground truth coefficient and the coefficient calculated from the predicted velocity and pressure field.

**Spearman's rank correlations for drag and lift coefficients** Given $K$ samples in the test set with the ground truth coefficients $C = \{C^1, \cdots, C^K\}$ (drag or lift) and the model predicted coefficients $\widehat{C} = \{\widehat{C}^1, \cdots, \widehat{C}^K\}$, the Spearman correlation coefficient is defined as the Pearson correlation coefficient between the rank variables, that is:

$$\rho = \frac{\mathrm{cov}\left(R(C)R(\widehat{C})\right)}{\sigma_{R(C)}\sigma_{R(\widehat{C})}}, \tag{16}$$

where $R$ is the ranking function, $\mathrm{cov}$ denotes the covariance and $\sigma$ represents the standard deviation of the rank variables. Thus, this metric is highly correlated to the model guide for design optimization. A higher correlation value indicates that it is easier to find the best design following the model-predicted coefficients (Spearman, 1961).

### B.3. Implementations

As shown in Table 8, all the baselines are trained and tested under the same training strategy. As for Transolver, the number of channels $C$ is set as 256 for high-dimensional inputs: Navier-Stokes, Shape-Net Car and AirfRANS, and 128 for the others. Further to balance efficiency, we set the number of slices $M$ as 32 for $C = 256$ configuration and 64 for $C = 128$. These configurations can also align our model parameters and running efficiency with other Transformer operators. Especially, we configure $\mathrm{Project}()$ in Eq. (1) as a single Linear layer for unstructured meshes: Elasticity, ShapeNet Car and AirfRANS, a convolution layer with $3 \times 3$ kernel for others. Next, we will present the implementation of all the baselines.

**Typical neural operators** All of these baselines have been widely examined in previous papers. Thus, for FNO and geo-FNO, we report their results following their official papers (Li et al., 2021; 2022). As for the other baselines, we follow the results in LSM (Wu et al., 2023). Note that all the other baselines expect geo-FNO cannot handle the unstructured mesh. Thus, their performances in Elasticity are evaluated by employing the special transformation in geo-FNO at the model

*Table 8.* Training and model configurations of Transolver. Training configurations are directly from previous works without extra tuning (Bonnet et al., 2022; Hao et al., 2023; Deng et al., 2024). Here $\mathcal{L}_{\rm v}$ and $\mathcal{L}_{\rm s}$ represent the loss on volume and surface fields respectively. As for Darcy, we adopt an additional spatial gradient regularization term $\mathcal{L}_{\rm g}$ following ONO (Xiao et al., 2024).

| Benchmarks | Training Configuration (Shared in all baselines) | | | | | Model Configuration | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Loss | Epochs | Initial LR | Optimizer | Batch Size | Layers $L$ | Heads | Channels $C$ | Slices $M$ |
| Elasticity | Relative L2 | 500 | $10^{-3}$ | AdamW (2019) | 1 | 8 | 8 | 128 | 64 |
| Plasticity | | | | | 8 | | | 128 | 64 |
| Airfoil | | | | | 4 | | | 128 | 64 |
| Pipe | | | | | 4 | | | 128 | 64 |
| Navier–Stokes | | | | | 2 | | | 256 | 32 |
| Darcy | $\mathcal{L}_{\rm rL2} + 0.1\mathcal{L}_{\rm g}$ | | | | 4 | | | 128 | 64 |
| Shape-Net Car | $\mathcal{L}_{\rm v} + 0.5\mathcal{L}_{\rm s}$ | 200 | $10^{-3}$ | Adam (2015) | 1 | 8 | 8 | 256 | 32 |
| AirfRANS | $\mathcal{L}_{\rm v} + \mathcal{L}_{\rm s}$ | 400 | | | 1 | | | 256 | 32 |

beginning and ending layer. However, we find that geo-FNO degenerates seriously in practical design tasks (Shape-Net Car and AirfRANS) even with comprehensively searched hyperparameters, that is number of layers in $\{2, 4, 6, 8\}$, number of channels in $\{128, 256, 512\}$ and number of Fourier basis in $\{16, 32, 64\}$. This may come from that geo-FNO is based on the periodic boundary assumption, while these two unstructured-mesh benchmarks apparently present complex boundaries. Since other neural operators are based on geo-FNO for unstructured meshes, we do not test them in practical design tasks.

**Transformer-based models** GNOT (Hao et al., 2023) reports the performance of itself and OFormer (Li et al., 2023c), Galerkin Transformer (Cao, 2021) in part of six standard benchmarks, which also adopts a comprehensive hyperparameter search for themselves and these two baselines. Thus, we report their results based on their official paper and results from GNOT. As for the other untested benchmarks, we search the hyperparameters as follows: number of layers in $\{2, 3, 4, 5, 6, 7, 8\}$, number of channels in $\{128, 256, 512\}$, number to heads in $\{1, 2, 4, 6, 8\}$. As for HT-Net (Liu et al., 2022) and FactFormer (Li et al., 2023d), since they employ the square window and axial decomposition respectively, they are inapplicable to Elasticity. Thus, we evaluate it under the aforementioned hyperparameter search on the other baselines. As for the ONO (Xiao et al., 2024) that is in submission, we adopted their official code provided in the OpenReview and rerun it under the same training and hyperparameter-search strategy as other baselines and also tried different linear attention designs (Katharopoulos et al., 2020a; Kitaev et al., 2020; Xiong et al., 2021; Cao, 2021; Choromanski et al., 2021a).

Experimentally, we found that ONO (Xiao et al., 2024) and OFormer (Li et al., 2023c) come across the unstable training problem on Shape-Net Car and AirfRANS. This may come from that ONO adopts the Cholesky decomposition to the channel attention map to ensure feature orthogonality, which requires the channel attention to be positive semidefinite. However, in large-scale mesh scenarios, this assumption may not be satisfied, making the model cannot be successfully executed. As for OFormer, it utilizes the cross-attention mechanism with target mesh points as queries and learned features as keys and values. However, directly calculating the attention among massive mesh points (over 32,000) is hard to optimize, causing the training loss curve to keep jittering. Thus, we do not report their performance in practical design tasks.

**Geometric deep models** We implement simple MLP, GraphSAGE (Hamilton et al., 2017), PointNet (Qi et al., 2017) and Graph U-Net (Gao & Ji, 2019) following the code base of AirfRANS (Bonnet et al., 2022). As for GNO (Li et al., 2020a) and 3D-GeoCA (Deng et al., 2024), we adopt the official code base of 3D-GeoCA. And we implement GINO based on its official code. Note that in the official paper, GINO is only trained to estimate the surface pressure of cars, which is not enough to calculate the drag coefficient. To ensure a comprehensive evaluation, we follow the experiment setting of 3D-GeoCA, which is to predict the surface pressure and surrounding air velocity simultaneously.

## C. Full Ablations

We include the complete ablations here as a supplement to Table 4 in the main text.

**Number of slices $M$** As demonstrated in Table 9, increasing the number of slices $M$ will generally boost the model performance. Also, it is notable that increasing $M$ will also bring extra computation costs. Too many slice tokens may also lead the attention calculation to potential noise or distraction, which could bring performance fluctuation or drop, such as in $M = 1024$ cases of Elasticity and Navier-Stokes. In this paper, we set $M$ as 64 for the models with 128 hidden channels and 32 for models with 256 hidden channels, which can accomplish a better balance between performance and efficiency.

*Table 9.* Full ablations on Physics-Attention. We experiment on three variants: changing the number of slices $M$ and replacing the learnable slices with fixed regular squares in the shape of $4 \times 4$ (#Regular Squares). Efficiency is calculated on 1024 mesh points and batch size as 1. Since #Regular Squares cannot handle unstructured inputs, we omit its efficiency and performance on Elasticity.

| Ablations | | #Memory (GB) | #Time (s/epoch) | Relative L2 | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Elasticity | Plasticity | Airfoil | Pipe | Naiver-Stokes | Darcy |
| Number of Slices | 1 | 0.60 | 37.76 | 0.0148 | 0.0140 | 0.0084 | 0.0087 | 0.1511 | 0.0386 |
| | 8 | 0.60 | 37.82 | 0.0071 | 0.0028 | 0.0056 | 0.0040 | 0.1136 | 0.0096 |
| | 16 | 0.61 | 37.96 | 0.0067 | 0.0019 | 0.0057 | 0.0045 | 0.0958 | 0.0067 |
| | 32 | 0.62 | 38.00 | 0.0067 | 0.0015 | 0.0067 | 0.0042 | 0.0900 | 0.0063 |
| | 64 | 0.64 | 38.18 | 0.0064 | 0.0012 | 0.0053 | 0.0033 | 0.0871 | 0.0059 |
| | 96 | 0.68 | 38.31 | 0.0061 | **0.0008** | 0.0054 | 0.0033 | 0.0802 | 0.0055 |
| | 128 | 0.69 | 42.24 | 0.0058 | 0.0009 | 0.0049 | 0.0034 | **0.0783** | 0.0054 |
| | 256 | 0.81 | 39.13 | **0.0054** | 0.0013 | **0.0043** | **0.0032** | 0.0856 | **0.0050** |
| | 512 | 1.01 | 39.75 | 0.0059 | 0.0012 | 0.0045 | 0.0040 | 0.0914 | 0.0056 |
| | 1024 | 1.53 | 40.49 | 0.0068 | 0.0017 | 0.0048 | 0.0047 | 0.1003 | 0.0055 |
| Regular Squares | | / | / | / | 0.0022 | 0.0071 | 0.0049 | 0.1077 | 0.0088 |

*Table 10.* Comparison between Plain Transformer (Vaswani et al., 2017) and Transolver at different resolutions of Darcy.

| Number of Mesh Points (Resolutions) | 484 (22×22) | 1,681 (41×41) | 3,364 (58×58) | 7,225 (85×85) | 10,609 (103×103) | 19,881 (141×141) | 44,521 (211×211) | 168,921 (411×411) |
|---|---|---|---|---|---|---|---|---|
| Plain Transformer | 0.02017 | 0.0103 | 0.0073 | 0.0081 | OOM | OOM | OOM | OOM |
| Transolver | 0.02019 | 0.0089 | 0.0058 | 0.0059 | 0.0057 | 0.0062 | 0.0063 | 0.0060 |
| Relative Promotion | -0.1% | 13.6% | 20.5% | 27.2% | / | / | / | / |

**Learnable slices or regular squares** In all benchmarks, using fixed regular squares will damage the model performance, even in the Navier-Stokes and Darcy which are originally discretized in the regular grid. This may result from the intricate and spatially deformed physics states in PDEs, further demonstrating the advantage of learning adaptive slices in Transolver.

**Comparison with full attention** To highlight the advantages of Transolver, we also compare it with Plain Transformer (Vaswani et al., 2017), which employs the full attention mechanism. Note that it is non-trivial to apply the plain Transformer to our benchmarks (1k-32k points). Even for the most powerful Large Language Model GPT-4 (Achiam et al., 2023), it can only handle up to 32k tokens, which is well-optimized and accelerated. When it comes to PDE-solving tasks, almost all of the previous Transformer-based models (Li et al., 2023c; Hao et al., 2023) utilize well-designed efficient attention for more than 1k mesh points instead of plain Transformer. Thus, the experiments of plain Transformer in over 1k tokens are with the help of the gradient-checkpointing technique (Chen et al., 2016), which can reduce the GPU memory but also significantly affect speed. Even so, we can only measure plain Transformer up to 7k tokens on one A100 40GB GPU.

Concretely, we downsample the Darcy dataset from 168,921 tokens ($411 \times 411$) to 484 tokens ($22 \times 22$) and replace Physics-Attention by canonical attention with identical architecture elsewhere. Note that directly downsampling the ground-truth data is unreasonable in the PDE context, since some physical interactions can only be observed in high resolution. That is why both models' performances drop seriously in 484 and 1,681 token settings. However, the relative promotions under different domain sizes are still referable. In Figure 8 of main text, we have demonstrated that Transolver performs stable in the range of 7,225 to 168,921 tokens. In the new experiments of Table 10, we can find that in the smallest resolution, Transolver performs similarly to plain Transformer, while the advantages of Transolver are getting more significant under larger mesh points, demonstrating the superiority of Physics-Attention design in handling large-scale meshes.

## D. Addition Visualizations

In this section, we provide more visualizations for learned slices and showcases as a supplement to Figure 5 and Figure 7.

### D.1. Learned Slices

**Original mesh** We visualize the learned slices on 5 benchmarks: Shape-Net Car (Figure 9), Airfoil (Figure 10), Pipe (Figure 12), Naiver-Stokes (Figure 14), and Darcy (Figure 16). These visualizations provide valuable insights into the

model's ability to capture diverse patterns and perceive subtle properties of physical states, including front-back-side pressure of driving cars, complex flow characteristics around the airfoil, fluid dynamics in pipes, swirling patterns of complex fluid interactions, and fluid-structure interactions along the porous medium. Especially, the model also learns a periodic diagonal pattern in the Naiver-Stokes equation, which corresponds to the periodic external force in the Navier-Stokes benchmark (Li et al., 2021). These findings demonstrate model's ability to learn underlying physical states behind complicated geometrics.

**Resampled mesh** To demonstrate that our design in learning physical states is free from concrete discretization, we also apply Transolver in resampled meshes in Figures 11-17, where we only keep 50%-80% mesh points of the original input. Note that this design may break the continuous and elaborately designed structure of the original mesh. Surprisingly, we can find that even for these broken meshes, Transolver can still capture physical states precisely, further verifying its geometry-general capability and highlighting the benefits of learning physical states.

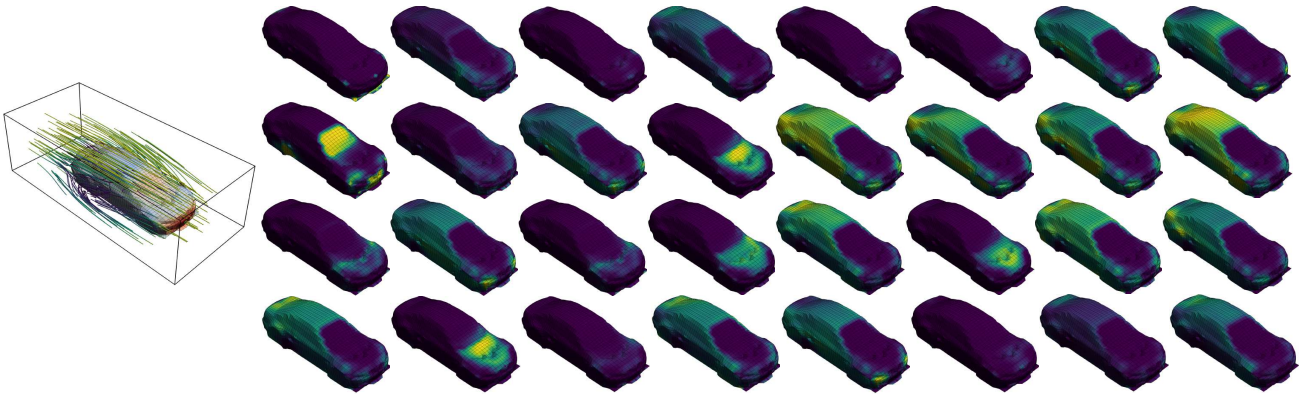


*Figure 9.* Visualization of learned slices on the Shape-Net Car benchmark (number of slices $M = 32$). Note that this benchmark involves both volume and mesh geometrics and velocity-pressure joint modeling. For clarity, we only present the slice weights on the surface here. Thus, the sum of visualized weights could be smaller than 1.

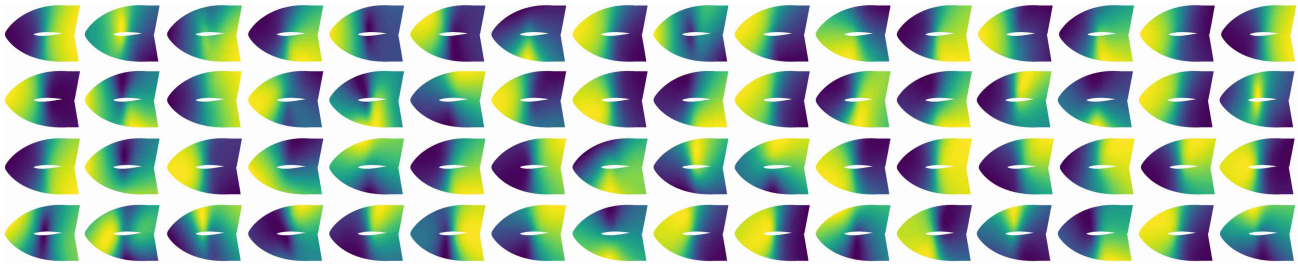


*Figure 10.* Visualization of learned slices on the original Airfoil benchmark (number of slices $M = 64$).

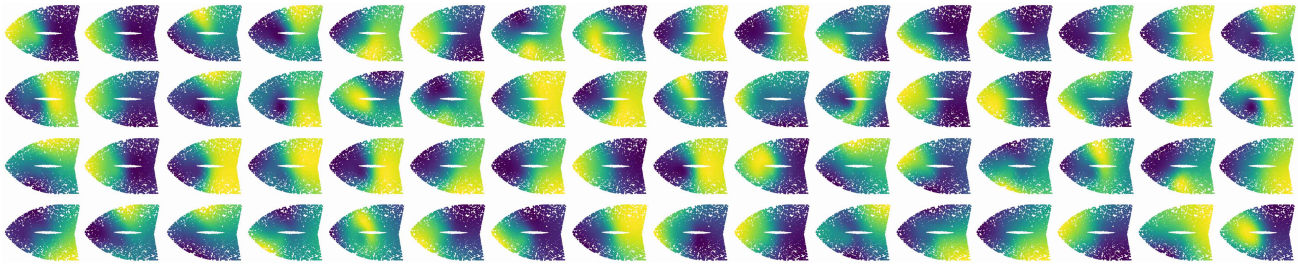


*Figure 11.* Visualization of learned slices on the randomly resampled Airfoil benchmark (number of slices $M = 64$).

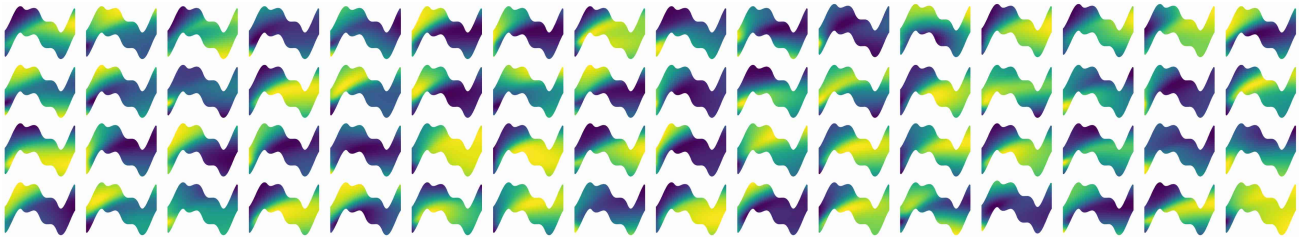


*Figure 12.* Visualization of learned slices on the original Pipe benchmark (number of slices $M = 64$).

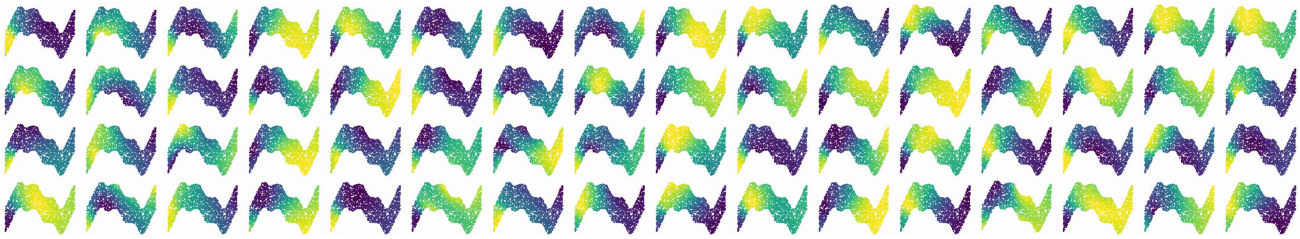


*Figure 13.* Visualization of learned slices on the randomly resampled Pipe benchmark (number of slices $M = 64$).

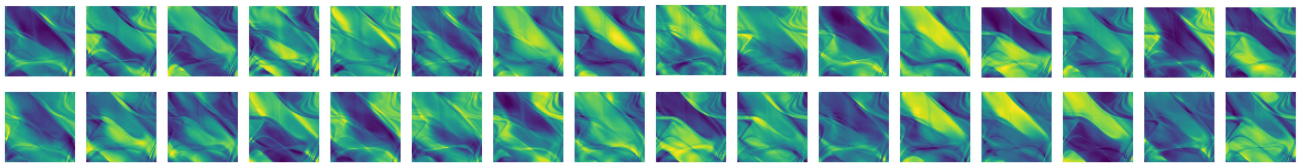


*Figure 14.* Visualization of learned slices on the original Navier-Stokes benchmark (number of slices $M = 32$).

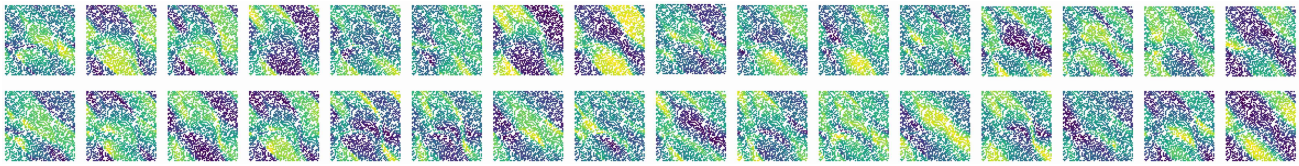


*Figure 15.* Visualization of learned slices on the randomly resampled Navier-Stokes benchmark (number of slices $M = 32$).

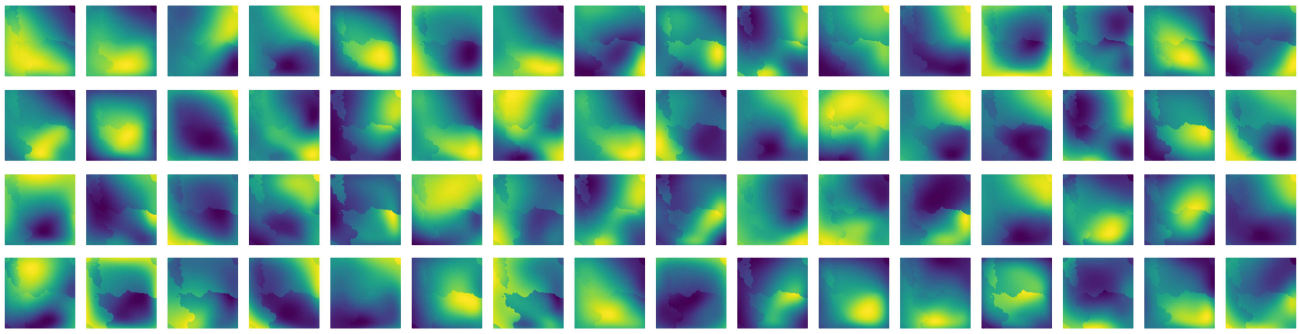


*Figure 16.* Visualization of learned slices on the original Darcy benchmark (number of slices $M = 64$).

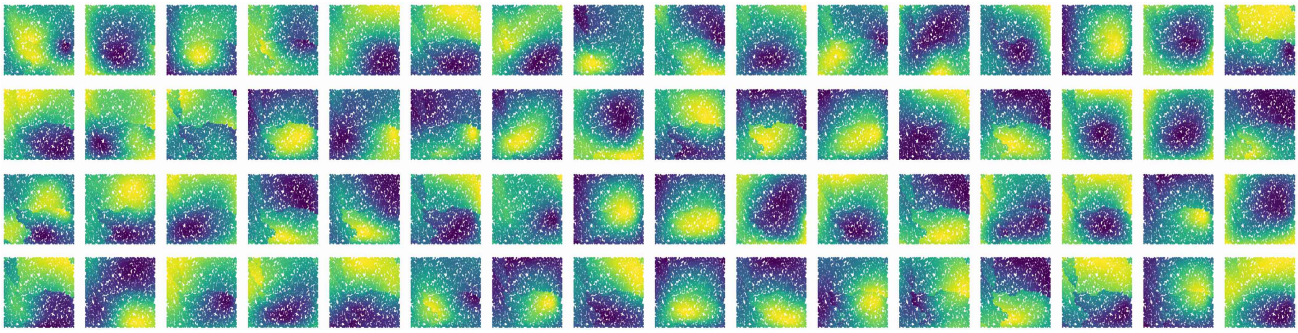


*Figure 17.* Visualization of learned slices on the randomly resampled Darcy benchmark (number of slices $M = 64$).

### D.2. Showcases

As shown in Figure 18, in comparison with the previous state-of-the-art models: LSM (Wu et al., 2023) and GNOT (Hao et al., 2023), Transolver excels in capturing the deformation of Plasticity, the shock wave of Airfoil, fluid in the Pipe end and the swirling parts of Navier-Stokes. Here, GNOT fails in predicting the future deformation of Plasticity.

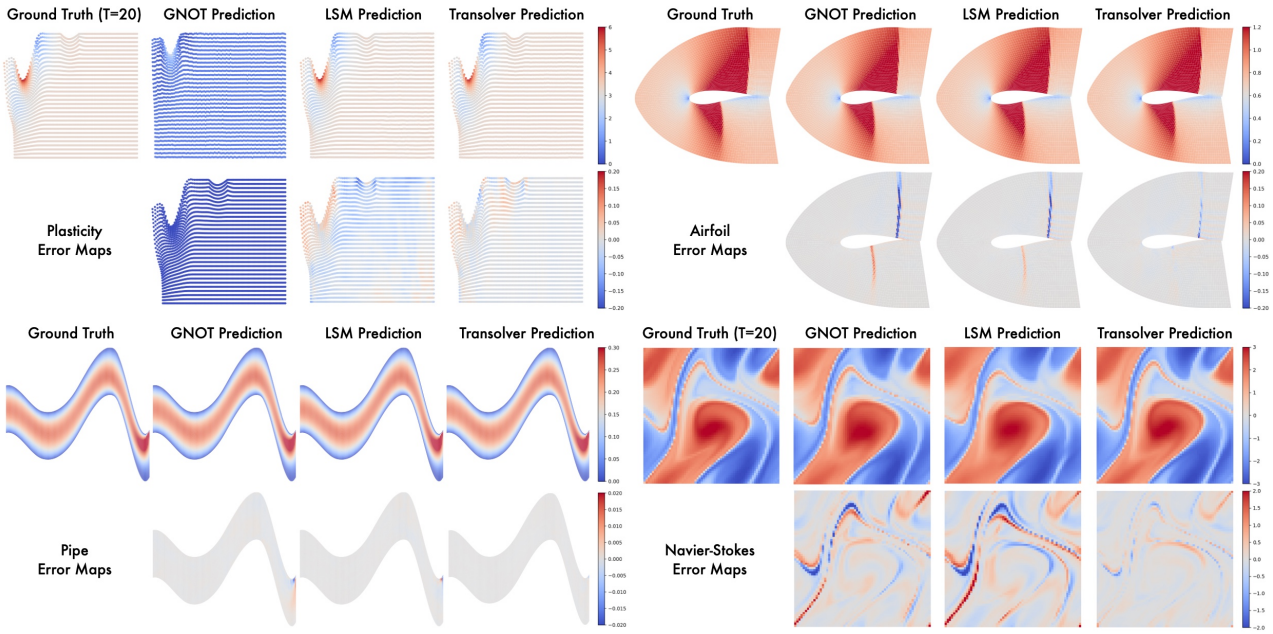



*Figure 18.* Showcase comparison with the previous best models: LSM and GNOT. Both prediction results and error maps are provided.

## E. Addition Experiments

Here we provide more experiments to complete the results of the main text and investigate new experiment settings.

### E.1. Model Scalability

In the main text, we have investigated the model scalability in Darcy regarding resolution, data and parameters. Here we provide the scalability experiments on more benchmarks. Since we only have the data generation code of the Darcy benchmark, we only test the parameter scalability for the other datasets. As shown in Figure 19, most of the benchmarks will benefit from a larger model size, especially for Elasticity (Relative L2: 0.0064 for 8 layers, 0.0047 for 40 layers) and Airfoil (Relative L2: 0.0053 for 8 layers and 0.0037 for 40 layers). These results highlight the scaling potential of Transolver. However, for Plasticity (Relative L2: 0.0012) and Pipe (Relative L2: 0.0033) whose performances are already close to saturation, increasing the model parameter will bring performance fluctuation. The slight performance drop may also come

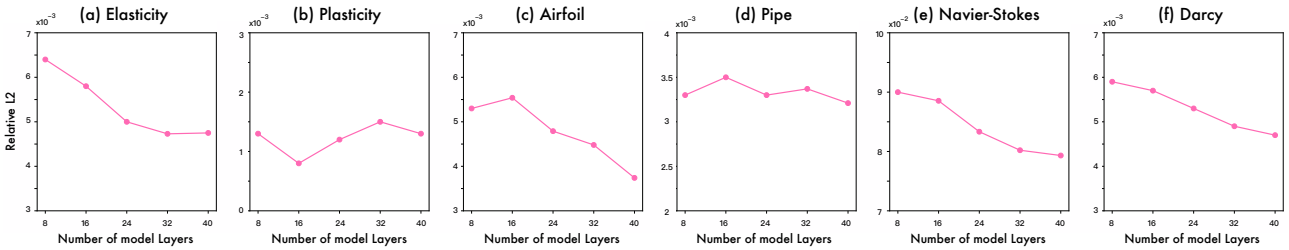



*Figure 19.* Parameter scalability on six standard benchmarks, where we gradually increase the number of model layers from 8 to 40.

from the stochasticity of optimization. Thus, a large dataset is expected to fully unlock the power of deep models in solving PDEs. Since we only focus on the model design in this paper, we would like to leave the data collection as future work.

### E.2. Adaptive Multiscale Modeling

Since $N$ mesh points are ascribed to $M$ slices as shown in Eq. (1), it is easy to calculate that the expectation of the number of mesh points per slice is $\frac{N}{M}$. Thus, we can control the granularity of physical state modeling by adjusting $M$, where a larger $M$ will derive more slices, leading to more fine-grained physical states. Further, in deep models, we can configure $M$ as different values in different layers to conveniently achieve multiscale modeling.

As shown in Figure 20, we further try the configuration that different layers are set with different numbers of slices. Since the calculation complexity of multiscale configuration is between $M = 64$ and $M = 32$, we also list the performance of these two official configurations for clarity in Table 11. Note that as we presented in Table 9, in general, increasing the number of slices $M$ will boost the model performance. But, surprisingly, we find that in some cases, the multiscale configuration can perform comparably or even better than the $M = 64$ official configuration. This may come from that both Darcy and Airfoil exhibit clear multiscale properties, thereby benefiting more from the hierarchical features.

*Table 11.* Comparison between official and multiscale configuration. Efficiency is calculated on 1024 mesh points and batch size as 1.

| Designs | #Mem (GB) | #Time (s/epoch) | Relative L2 | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | Elasticity | Plasticity | Airfoil | Pipe | Naiver-Stokes | Darcy |
| Official Config $M = 32$ | 0.62 | 38.00 | 0.0067 | 0.0015 | 0.0067 | 0.0042 | 0.0900 | 0.0063 |
| Official Config $M = 64$ | 0.64 | 38.18 | **0.0064** | **0.0012** | 0.0053 | **0.0033** | **0.0871** | 0.0059 |
| Multiscale Config | 0.62 | 38.10 | 0.0066 | **0.0012** | **0.0050** | 0.0037 | 0.0891 | **0.0056** |

Different from the well-acknowledged multiscale deep models, such as Swin Transformer (Liu et al., 2021) or U-Net (Ronneberger et al., 2015), our design in learning slices is not limited by the discretization. This means that we can set the number of slices $M$ at will, regardless of the exact division restriction. This also makes our model free from inflexible padding operations. In this paper, we mainly experiment with the official configuration. We would like to leave the exploration of the multiscale architecture of Transolver as a future work.

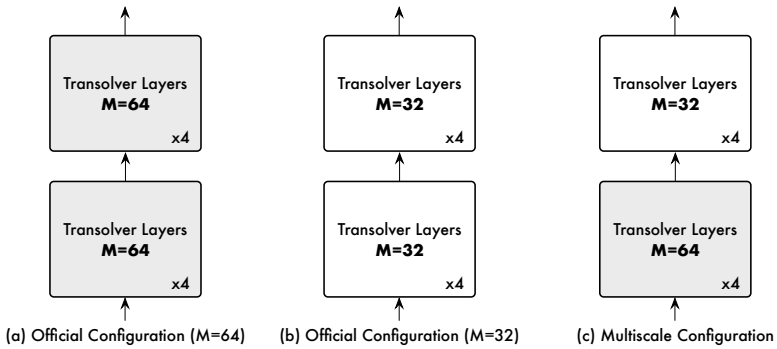



*Figure 20.* Illustration of adaptive multiscale modeling in Transolver. In addition to the official configuration that all Transolver layers share the same number of slices, we also provide the multiscale configuration where the last 4 layers only use half slices w.r.t first 4 layers.

*Table 12.* Settings of OOD generalization experiments on the AirfRANS. The range of Reynolds and angles are listed.

| Dataset | OOD Reynolds | | OOD Angles | |
|---|---|---|---|---|
| | Reynolds Range | Samples | Angles Range | Samples |
| Training set | $[3 \times 10^6, 5 \times 10^6]$ | 500 | $[-2.5°, 12.5°]$ | 800 |
| Test set | $[2 \times 10^6, 3 \times 10^6] \cup [5 \times 10^6, 6 \times 10^6]$ | 500 | $[-5°, -2.5°] \cup [12.5°, 15°]$ | 200 |

### E.3. OOD Generalization

In Table 6, we have provided experiments of out-of-distribution Reynolds and angles of attacks. As a supplement, we include the detailed settings in Table 12, where the training and test sets contain completely different Reynolds or angles of attacks.

We also include the complete results in Table 13. It is impressive that Transolver can still achieve the consistent best performance in the out-of-distribution settings. Specifically, the relative promotion of Transolver is more significant than the i.i.d. setting, further demonstrating Transolver's generalizability advantage. This may come from our special design in learning physical states, which enables Transolver to capture more foundational physics information.

*Table 13.* Performance comparison on out-of-distribution tasks of AirfRANS. Relative L2 of the surrounding (Volume) and surface (Surf) physics fields, the relative L2 of lift coefficient ($C_L$) is also recorded, along with Spearman's rank correlations $\rho_L$.

| Model | OOD Reynolds | | | | OOD Angles | | | |
|---|---|---|---|---|---|---|---|---|
| | Volume ↓ | Surf ↓ | $C_D$ ↓ | $\rho_D$ ↑ | Volume ↓ | Surf ↓ | $C_L$ ↓ | $\rho_L$ ↑ |
| Simple MLP | 0.0669 | 0.1153 | 0.6205 | 0.9578 | 0.1309 | 0.3311 | 0.4128 | 0.9572 |
| GraphSAGE (Hamilton et al., 2017) | 0.0798 | 0.1254 | 0.4333 | 0.9707 | 0.1192 | 0.2359 | 0.2538 | 0.9894 |
| PointNet (Qi et al., 2017) | 0.0838 | 0.1403 | 0.3836 | 0.9806 | 0.2021 | 0.4649 | 0.4425 | 0.9784 |
| Graph U-Net (Gao & Ji, 2019) | 0.0538 | 0.1168 | 0.4664 | 0.9645 | 0.0979 | 0.2391 | 0.3756 | 0.9816 |
| MeshGraphNet (Pfaff et al., 2021) | 0.2789 | 0.2382 | 1.7718 | 0.7631 | 0.4902 | 1.1071 | 0.6525 | 0.8927 |
| GNO (Li et al., 2020a) | 0.0833 | 0.1562 | 0.4408 | 0.9878 | 0.1626 | 0.2359 | 0.3038 | 0.9884 |
| Galerkin (Cao, 2021) | 0.0330 | 0.0972 | 0.4615 | 0.9826 | 0.0577 | 0.2773 | 0.3814 | 0.9821 |
| GNOT (Hao et al., 2023) | 0.0305 | 0.0959 | 0.3268 | 0.9865 | 0.0471 | 0.3466 | 0.3497 | 0.9868 |
| GINO (Li et al., 2023a) | 0.0839 | 0.1825 | 0.4180 | 0.9645 | 0.1589 | 0.2469 | 0.2583 | 0.9923 |
| **Transolver (Ours)** | **0.0143** | **0.0364** | **0.2996** | **0.9896** | **0.0357** | **0.2275** | **0.1500** | **0.9950** |

### E.4. Apply to Lagrangian Settings

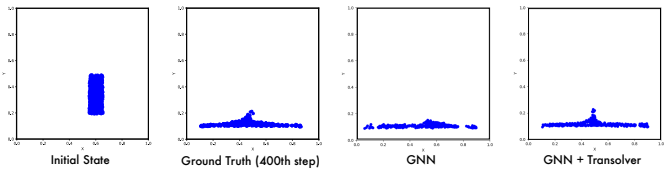
In the main text, we follow the convention of previous neural operators (Li et al., 2021; Wu et al., 2023) and experiment with Eulerian datasets, where the geometry of input data is fixed. There is another branch of tasks, named Lagrangian settings, which simulates the dynamics system (e.g. fluid) by tracking a series of particles. To further verify the effectiveness of Transolver in handling ever-changing geometrics, we also experiment with a Lagrangian PDE-solving task.

As for Lagrangian settings, the convention is to construct a graph at each timestep and utilize GNN-based models to capture local interactions among particles (Sanchez-Gonzalez et al., 2020). Although the capability to process irregular meshes also enables Transolver to receive the scattered particles as inputs, the essential graph information is missing. Thus, we did a preliminary experiment on the WaterDrop process (Sanchez-Gonzalez et al., 2020), whose task is to predict the future 994 steps based on the past 6 steps for 2,000 particles. Concretely, we enhance the GNN message passing with an additional Transolver layer in parallel. Both models are trained with 1M iterations, which requires 2 days in one A100 40GB GPU.

As shown in Table 14, Transolver can further boost the GNN performance by a sharp margin and generate a more accurate future, especially for the water splash process. This result verifies the benefits of Transolver in enhancing physics learning.

*Table 14.* Performance comparison on the Lagrangian WaterDrop dataset. Position MSE of 2,000 predicted particles is recorded.

| Model | MSE ↓ |
|---|---|
| GNN (Sanchez-Gonzalez et al., 2020) | 0.0182 |
| **GNN + Transolver (Ours)** | **0.0069** |
| Relative Promotion | 62.1% |



### E.5. Standard Deviations

We repeat all the experiments three times and provide standard deviations here. As shown in Table 15, Transolver surpasses the previous state-of-the-art models with high confidence. It is worth noticing that we compare Transolver with the second-best model on each benchmark. It is hard to outperform all the previous models consistently given that the previous models have shown ups and downs on different benchmarks, further verifying the effectiveness of our model.

Especially for AirfRANS (Bonnet et al., 2022) that contains diverse conditions on airfoil shape, Reynolds number and angle of attack, according to their official paper, they "choose to only run 1000 simulations as one of the goals of this dataset is to be close to real-world settings, i.e. limited quantity of data." Thus, this limited data setting will result in a relatively large deviation. Notably, even in this hard setting, Transolver still surpasses the second-best model with 95% confidence.

*Table 15.* Standard deviations of Transolver on all experiments. For clarity, we also list the performance of the second-best model. Especially, for Shape-Net Car and AirfRANS, standard deviations on Spearman's rank correlations of drag or lift coefficients are provided.

| Model ($\times 10^{-2}$) | Point Cloud | Structured Mesh | | | Regular Grid | | Unstructured Mesh | |
|---|---|---|---|---|---|---|---|---|
| | Elasticity | Plasticity | Airfoil | Pipe | Navier–Stokes | Darcy | Shape-Net Car | AirfRANS |
| Second-best Model | 0.86±0.02 (GNOT) | 0.17±0.01 (OFormer) | 0.59±0.01 (LSM) | 0.47±0.02 (GNOT) | 11.95±0.20 (ONO) | 0.65±0.01 (LSM) | 98.42±0.12 (3D-GeoCA) | 99.64±0.07 (GraphSAGE) |
| Transolver | **0.64**±0.02 | **0.12**±0.01 | **0.53**±0.01 | **0.33**±0.02 | **9.00**±0.13 | **0.57**±0.01 | **99.35**±0.10 | **99.78**±0.04 |
| Confidence Interval | 99% | 99% | 99% | 99% | 99% | 99% | 99% | 95% |

## F. Full Efficiency Analysis

As a supplement to Figure 6 in the main text, we also conduct experiments on different sizes of input meshes and record the model parameter, running time and the GPU memory of five Transformer-based methods, which are Transolver, ONO, GNOT, OFormer, Galerkin. From Figure 21 and Table 16, we can find that in comparison with other methods, Transolver presents the least running time growth, which benefits from our design of linear-complexity Physics-Attention. Especially in large-scale meshes, Transolver is nearly 5x times faster than method ONO with 23% the GPU memory usage.

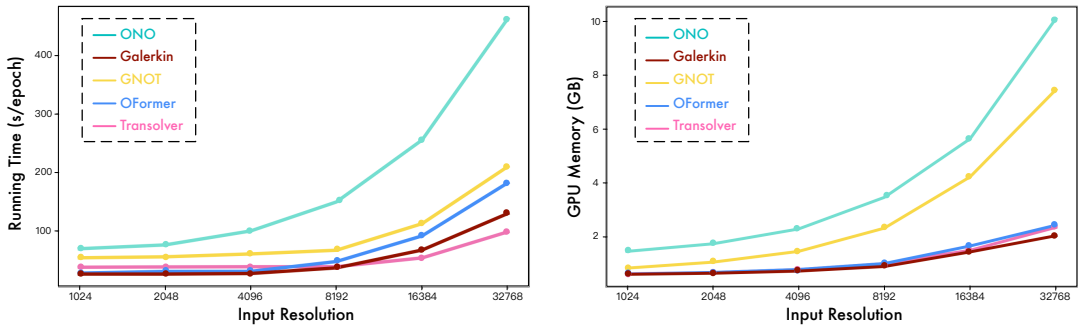



*Figure 21.* The growth curves of Transformer-based models w.r.t. the size of input mesh, where the batch size is set as 1. We record the running time and the GPU memory under different mesh points $N$, which range from $2^{10}$ to $2^{15}$.

*Table 16.* Model efficiency comparison in Elasticity (Relative L2) and Shape-Net Car ($\rho_D$), where we select five Transformer-based methods that can be applied to unstructured meshes. Efficiency is evaluated on inputs of different meshes during training. Running time is measured by the time to complete one epoch, which contains $10^3$ iterations. "/" indicates the baseline will fail in this benchmark.

| Input Mesh Size ($N$) | | Parameter | GPU Memory | Running Time | Elasticity | Shape-Net Car |
|---|---|---|---|---|---|---|
| Model | $N$ | (MB) | (GB) | (s / epoch) | (972 mesh points) | (32,186 mesh points) |
| **Transolver (Ours)** | 1024 | 0.9284 | 0.64 | 38.183 | 0.0064 | 0.9935 |
| | 2048 | 0.9284 | 0.69 | 38.678 | | |
| | 4096 | 0.9284 | 0.80 | 39.011 | | |
| | 8192 | 0.9284 | 1.02 | 39.048 | | |
| | 16384 | 0.9284 | 1.51 | 54.104 | | |
| | 32768 | 0.9284 | 2.36 | 98.552 | | |
| GNOT (Hao et al., 2023) | 1024 | 5.2477 | 0.85 | 54.282 | 0.0086 | 0.9833 |
| | 2048 | 5.2477 | 1.07 | 55.939 | | |
| | 4096 | 5.2477 | 1.47 | 60.857 | | |
| | 8192 | 5.2477 | 2.33 | 67.170 | | |
| | 16384 | 5.2477 | 4.23 | 112.552 | | |
| | 32768 | 5.2477 | 7.46 | 209.923 | | |
| ONO (Xiao et al., 2024) | 1024 | 1.1093 | 1.47 | 69.759 | 0.0118 | / |
| | 2048 | 1.1093 | 1.75 | 76.245 | | |
| | 4096 | 1.1093 | 2.30 | 100.134 | | |
| | 8192 | 1.1093 | 3.47 | 149.598 | | |
| | 16384 | 1.1093 | 5.64 | 255.339 | | |
| | 32768 | 1.1093 | 10.09 | 462.459 | | |
| OFormer (Li et al., 2023c) | 1024 | 0.8844 | 0.63 | 28.147 | 0.0183 | / |
| | 2048 | 0.8844 | 0.69 | 30.983 | | |
| | 4096 | 0.8844 | 0.80 | 31.113 | | |
| | 8192 | 0.8844 | 1.02 | 47.904 | | |
| | 16384 | 0.8844 | 1.67 | 91.671 | | |
| | 32768 | 0.8844 | 2.44 | 182.205 | | |
| Galerkin Transformer (Cao, 2021) | 1024 | 1.0414 | 0.62 | 26.507 | 0.0240 | 0.9764 |
| | 2048 | 1.0414 | 0.66 | 26.503 | | |
| | 4096 | 1.0414 | 0.74 | 27.481 | | |
| | 8192 | 1.0414 | 0.91 | 37.098 | | |
| | 16384 | 1.0414 | 1.45 | 67.524 | | |
| | 32768 | 1.0414 | 2.05 | 129.872 | | |