Practical Neural PDE Solvers: Complex Geometries & OOD Generalization

Haixu Wu

School of Software, Tsinghua University

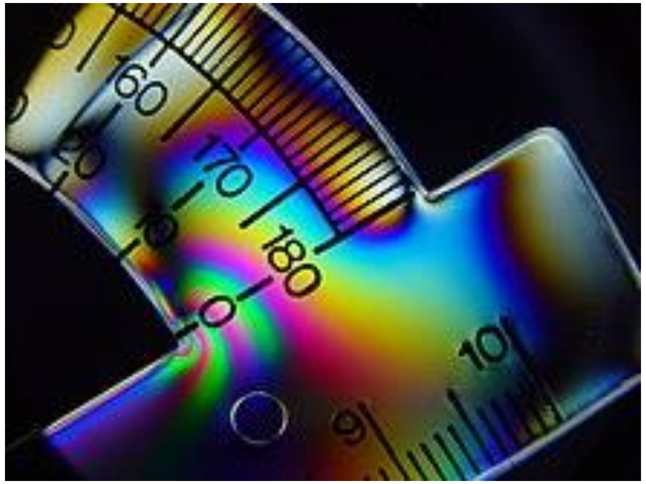
Oct 10, 2024

清華大學
Tsinghua University

Real-world phenomena

Turbulence

Atmospheric circulation



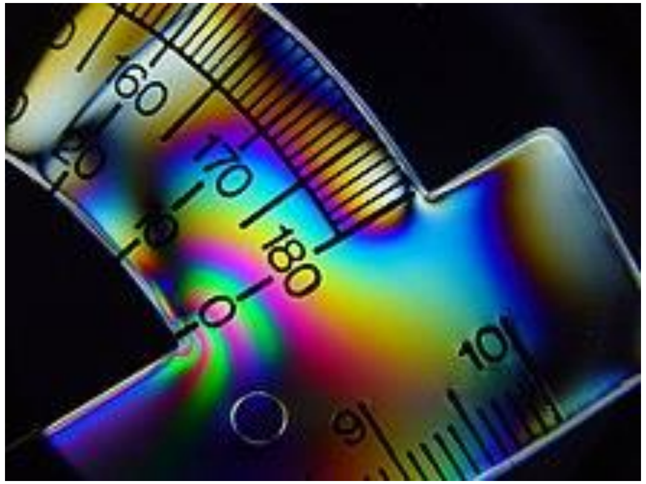
Stress

How to understand the world?

Real-world phenomena

Turbulence

Atmospheric circulation



Stress

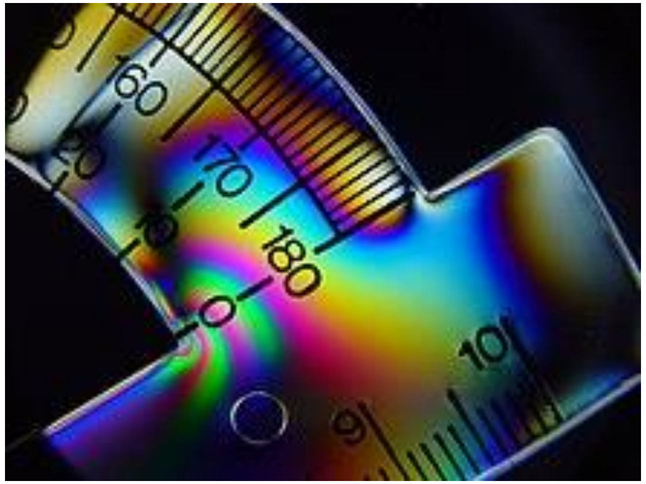
How to understand the world?

Images? Videos?

Real-world phenomena


Turbulence


Atmospheric circulation


Stress

Beyond appearances, these phenomena are governed by **scientific rules**.

Partial Differential Equations (PDEs)

- Fluid physics:

Navier-Stokes Equation
for fluid dynamics

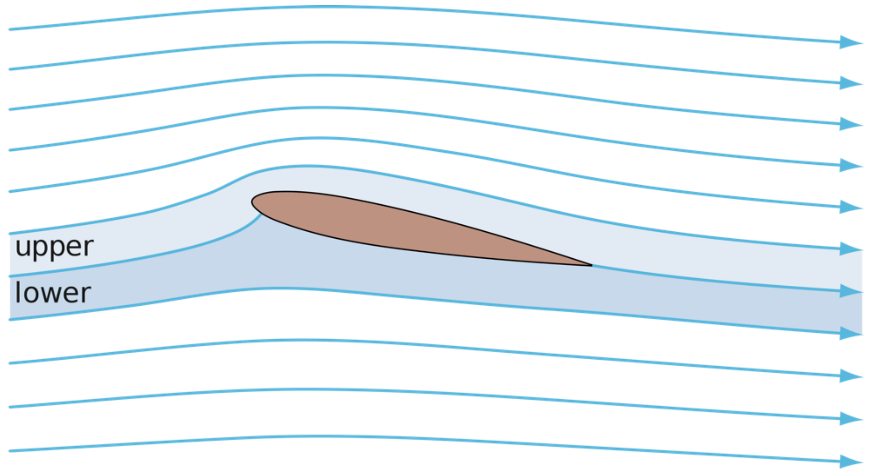
$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}) = 0$$
$$\frac{\partial \boldsymbol{U}}{\partial t} + \boldsymbol{U} \cdot \nabla \boldsymbol{U} = \boldsymbol{f} + \frac{1}{\rho} \nabla \cdot (\boldsymbol{T}_{ij} \boldsymbol{e}_i \boldsymbol{e}_j)$$
$$\frac{\partial (e + \frac{1}{2} \boldsymbol{U}^2)}{\partial t} + \boldsymbol{U} \cdot \nabla (e + \frac{1}{2} \boldsymbol{U}^2) = \boldsymbol{f} \cdot \boldsymbol{U} + \frac{1}{\rho} \nabla \cdot (\boldsymbol{U} \cdot \boldsymbol{T}_{ij} \boldsymbol{e}_i \boldsymbol{e}_j) + \frac{\lambda}{\rho} \Delta T$$

- Solid physics:

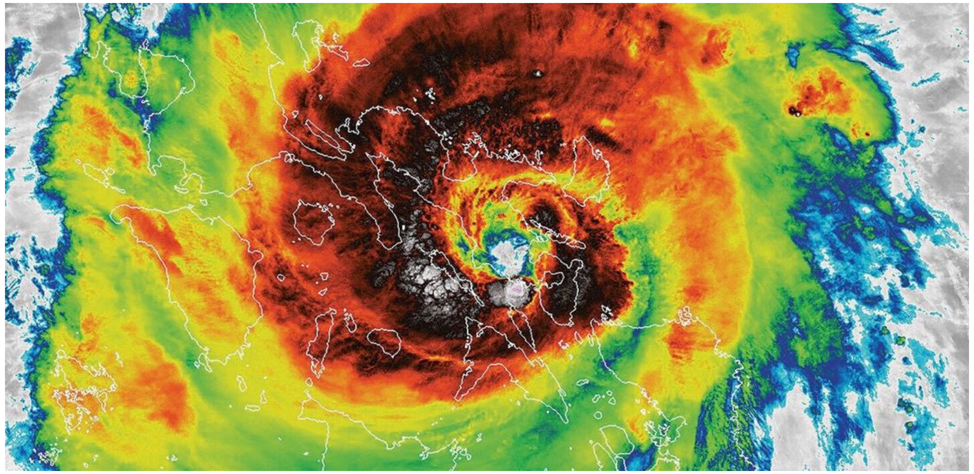
$$\rho^s \frac{\partial^2 \boldsymbol{u}}{\partial t^2} + \nabla \cdot \boldsymbol{\sigma} = 0$$

Inner stress
of solid materials

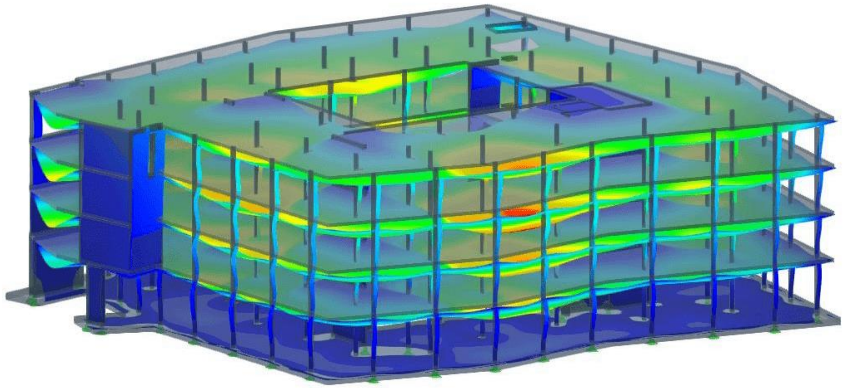
Wide Applications



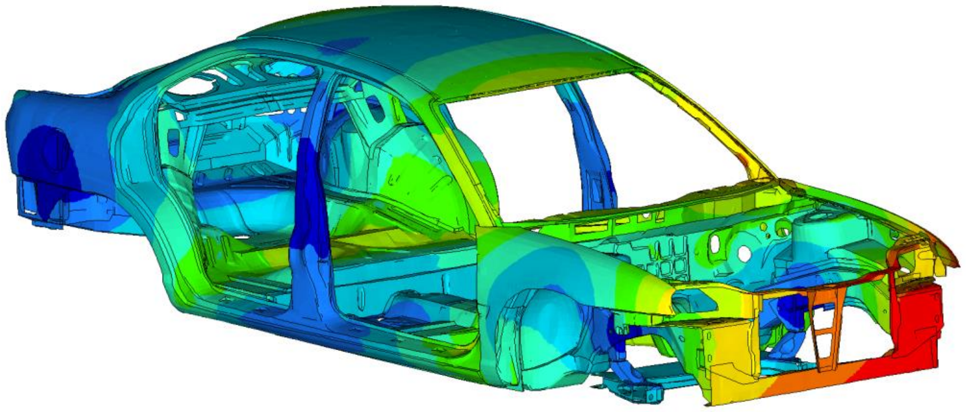
Airfoil design



Weather forecasting



Civil engineering

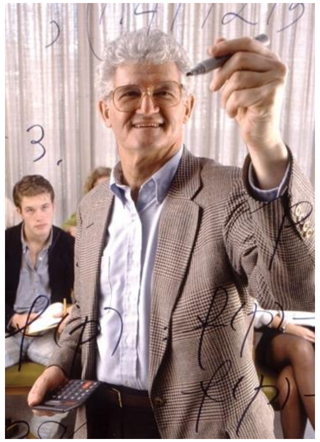


Vehicle manufacturing

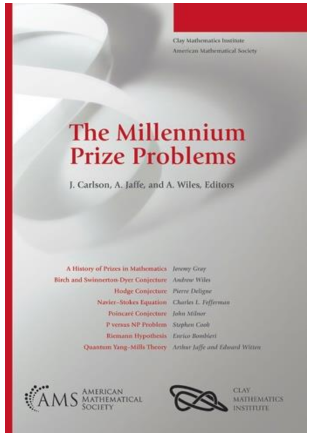
Difficulties in Solving PDEs


David Hilbert


John von Neumann


Peter Lax


Richard Courant



Millennium Prize Problems

- Birch and Swinnerton-Dyer conjecture
- Hodge conjecture
- **Navier–Stokes existence and smoothness**
- **P versus NP problem**
- Riemann hypothesis
- Yang–Mills existence and mass gap
- Poincaré conjecture (Solved)

It is really hard (usually impossible) to obtain the analytic solution of PDEs

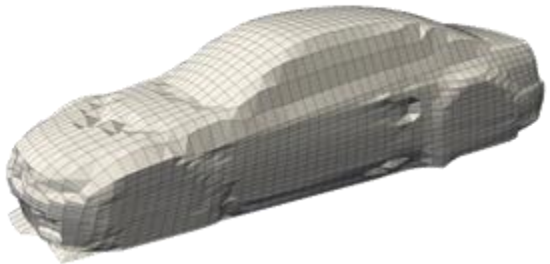
PDE Solvers

Classic Numerical Methods

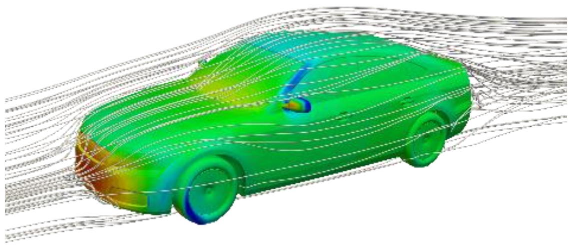
New Task → FEM, Spectral, etc → Results

- Recalculation for every new sample
- Each round will take huge costs

Stable but Slow



⇩ Days or even Months



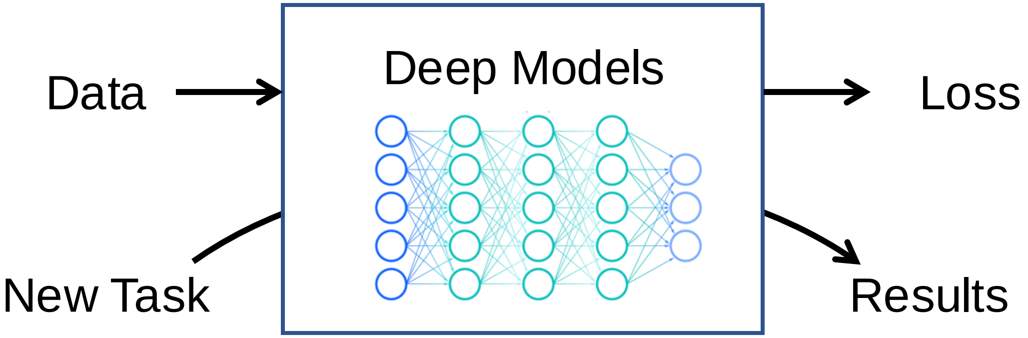
PDE Solvers

Classic Numerical Methods

- Recalculation for every new sample
- Each round will take huge costs

Stable but Slow

Neural PDE Solver



- Training once, inference a lot
- Each round needs several seconds

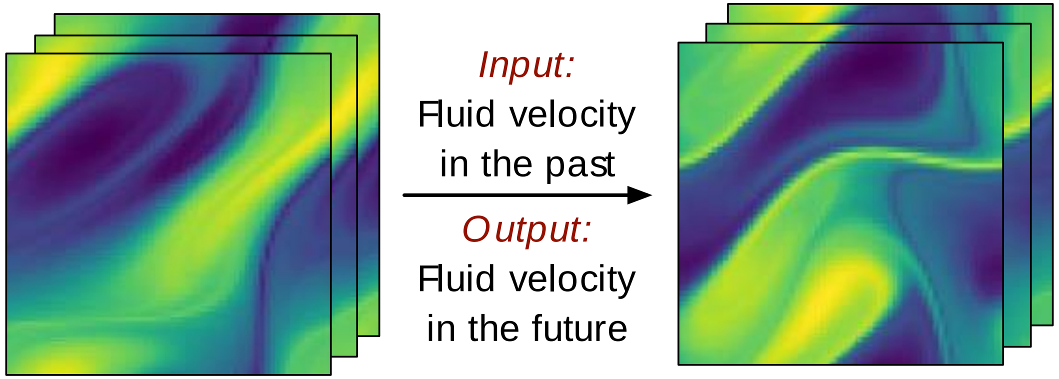
An efficient surrogate tool

(In expectation)

Challenges for Neural PDE Solvers

Most of Previous Neural PDE Solvers

Toy Problems & Small Models & Limited Diversity

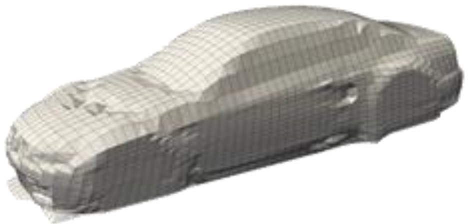


- 64×64 Inputs
- Less than 1M Parameters
- Fixed Viscosity and Boundary

Challenges for Neural PDE Solvers

Practical Applications:

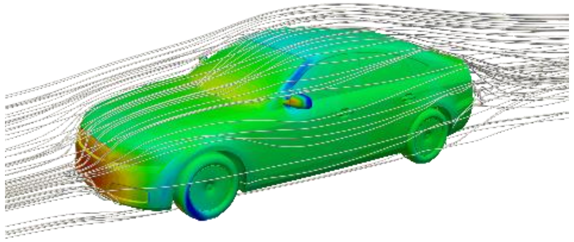
Large-scale Meshes & Diverse Applications



Varied Geometries

Ansys DASSAULT SYSTEMES

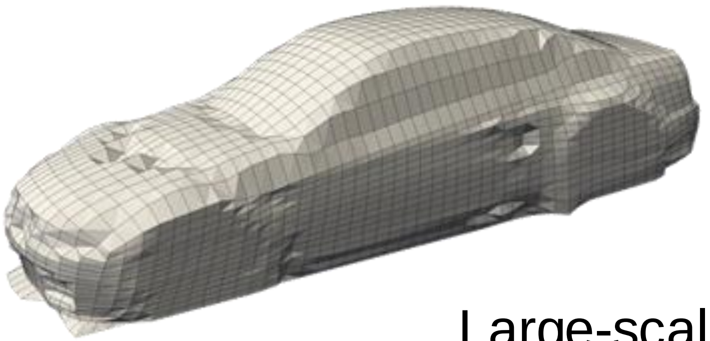
SIEMENS HEXAGON



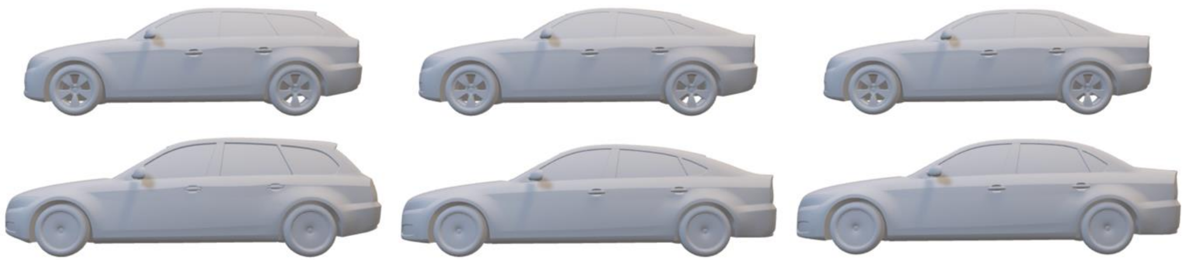
Physical Simulation

We need practical neural solvers for <u>large-scale meshes</u> and <u>diverse PDEs</u>

Our Exploration for Practical Neural PDE Solvers



Large-scale Meshes



Diverse PDEs, e.g. boundaries, coefficients, forces

1. Foundation Backbone:

   Transolver

2. Generalizable Model:

   Unisolver

ICML | 2024 Spotlight

The Forty-first International Conference on Machine Learning

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]

Haixu Wu

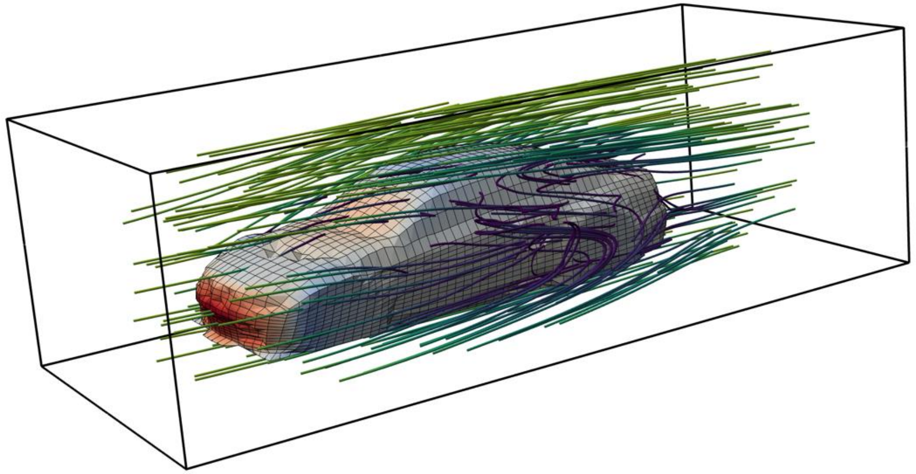
Huakun Luo

Haowen Wang

Jianmin Wang

Mingsheng Long

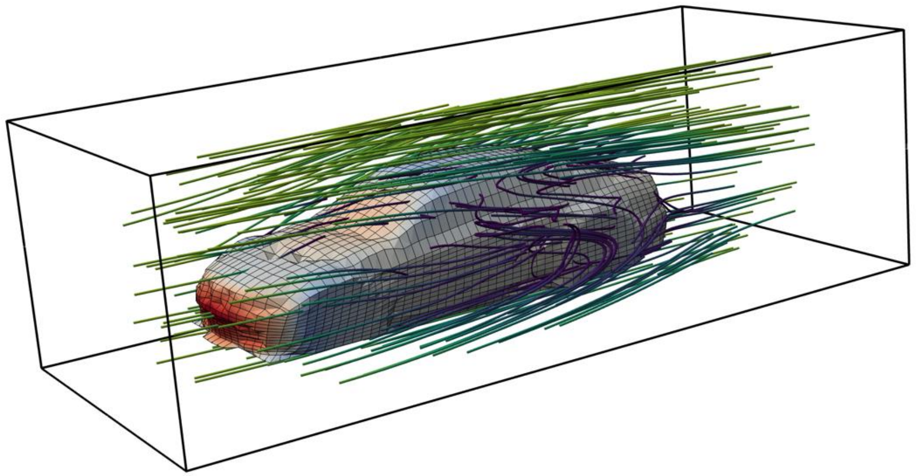
Challenges in Practical Industrial Design



Example: Estimate the drag coefficient of a given shape:

Surrounding Wind & Surface Pressure

Challenges in Practical Industrial Design

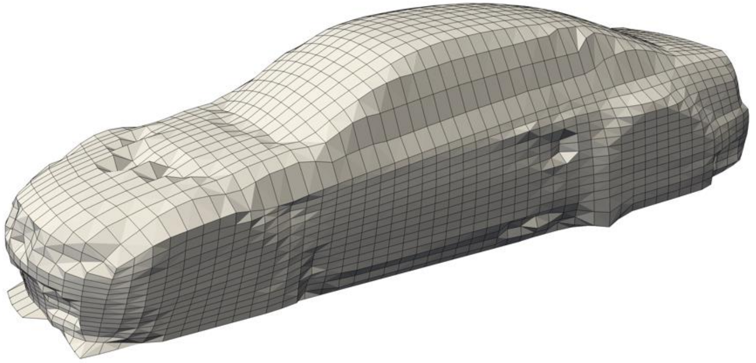


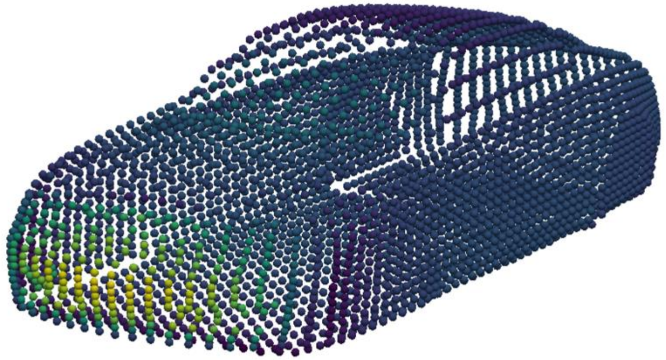
Example: Estimate the drag coefficient of a given shape:

Surrounding Wind & Surface Pressure

1. Large-scale meshes → **Huge computation cost**
2. Complex and unstructured geometrics → **Complex geometric learning**
3. Multiphysics interaction → **Intricate physical correlations**

Previous Work: Geometric Deep Learning





(1) Mesh

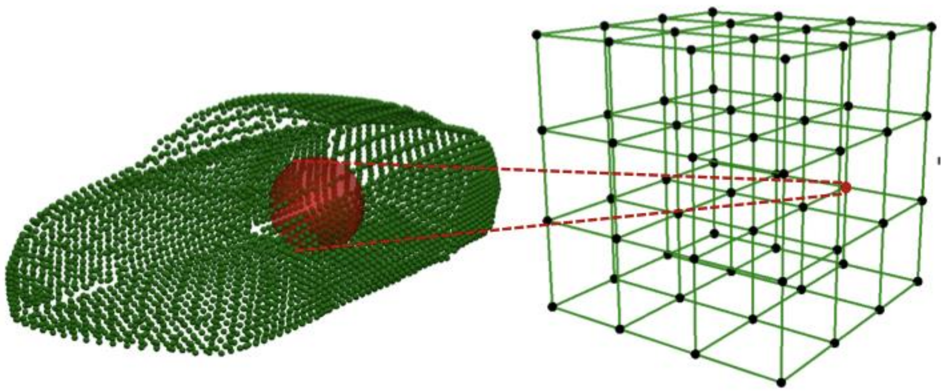
GraphSAGE, MeshGraphNet, etc

(2) Point Cloud

PointNet, Point Transformer, etc

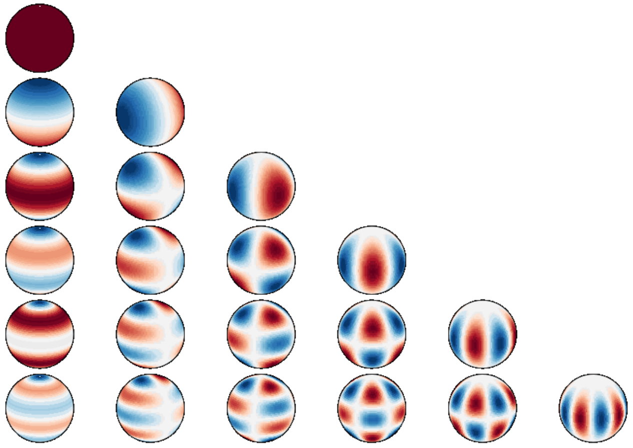
Excels in geometry modeling but fail in physics learning

Previous Work: Geometry-General Neural Operators



(1) GNN as Operators

GNO, GINO, etc

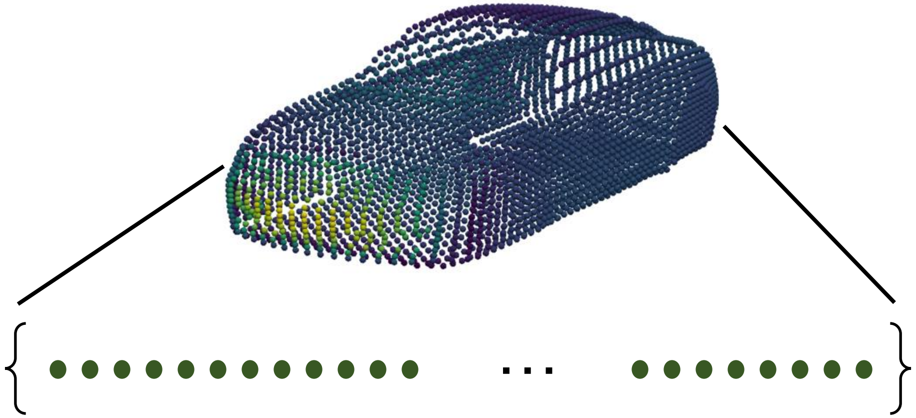


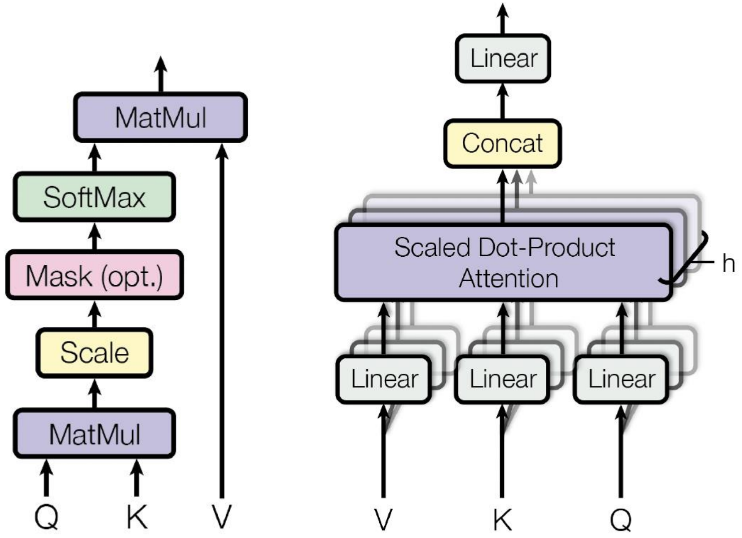
(2) FNO-Variants

geoFNO, SFNO, etc

Only focus on local physics or limited to periodic boundary

Transformer-based PDE Solvers





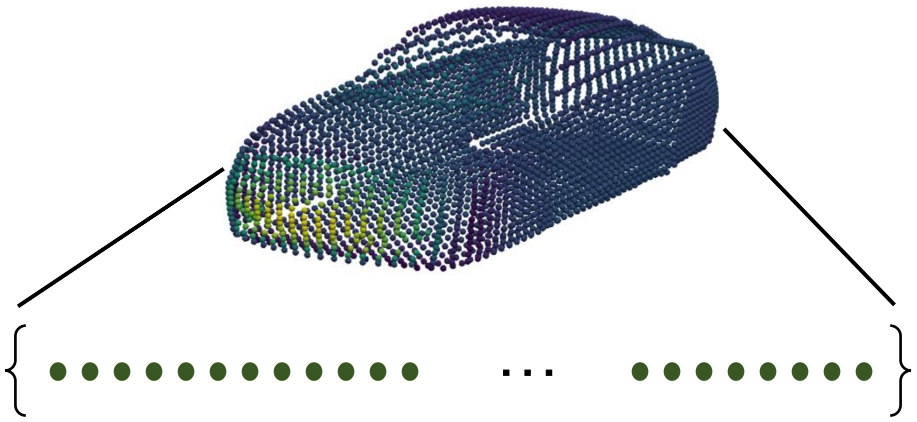
(1) Geometries as point sequences (2) Attention as Monte Carlo Integral

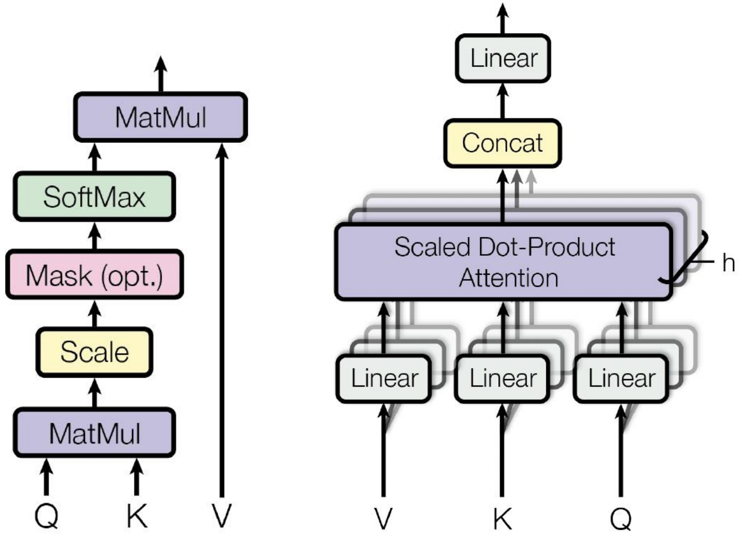
OFormer, Galerkin Transformer, etc

1. **Quadratic complexity (The effective length of GPT-4 is only 64k)**
2. **Hard to capture physical correlations among massive points**

Hsieh et al. RULER: What's the Real Context Size of Your Long-Context Language Models? COLM 2024

Transformer-based PDE Solvers





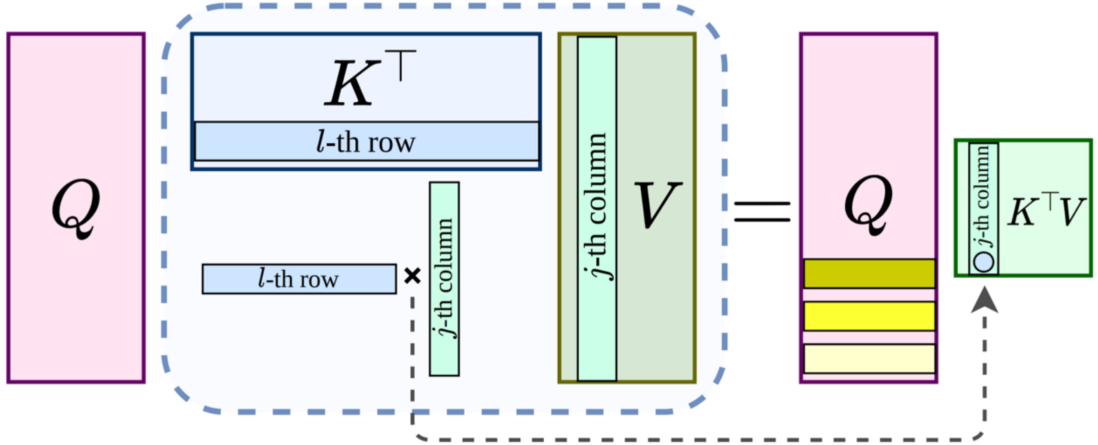
(1) Geometries as point sequences (2) Attention as Monte Carlo Integral

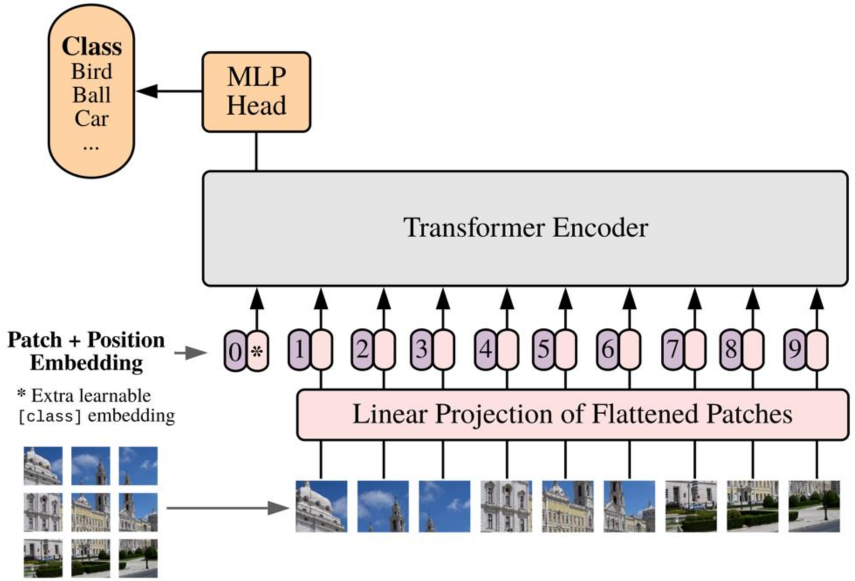
OFormer, Galerkin Transformer, etc

How to efficiently capture physical correlations underlying discretized meshes

is the key to "transform" Transformers into practical PDE solvers

Related Work





(1) Linear Transformers

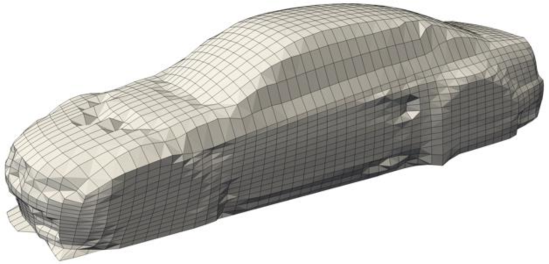
1. *Distracted attention*
2. *Individual points is insufficient for physics learning*

(2) Vision Transformer

Augment features with patch ✓

Not applicable to irregular meshes

A foundational Idea of Transolver



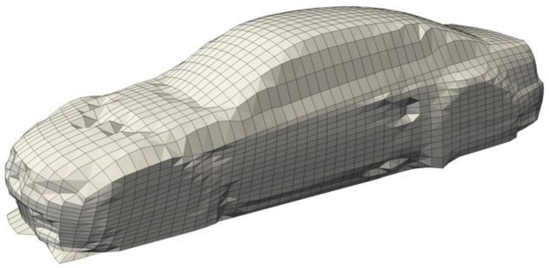
Discretized Domain

Previous Work

Being “trapped” to superficial and unwieldy meshes

Difficulties in Complexity, Geometry, Physics

A foundational Idea of Transolver

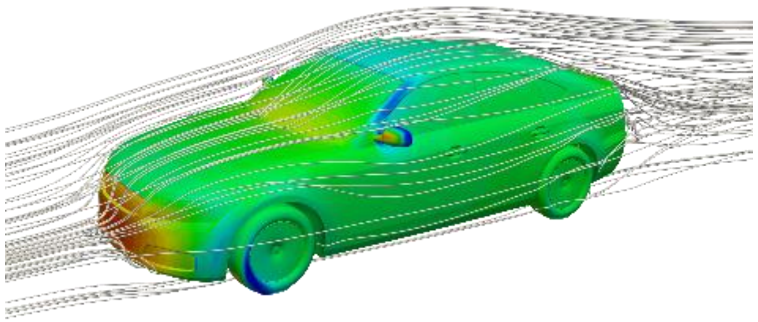


Discretized Domain

Previous Work

Being “trapped” to superficial and unwieldy meshes

Difficulties in Complexity, Geometry, Physics



Physics Domain

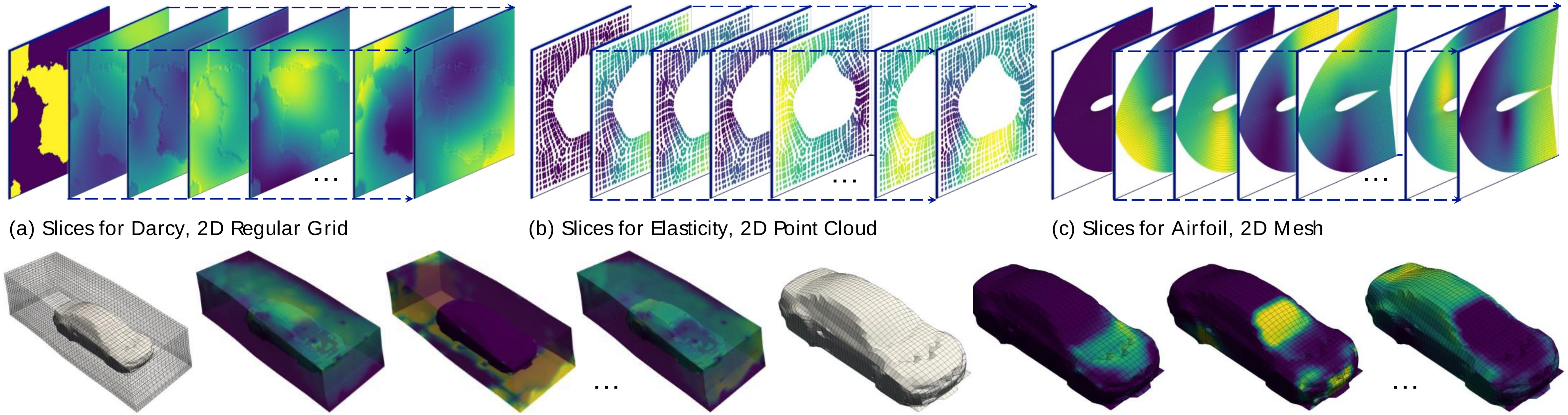
Transolver

Learning **intrinsic physical states** under

complex and large-scale geometrics

Better Complexity, Geometry, Physics Modeling

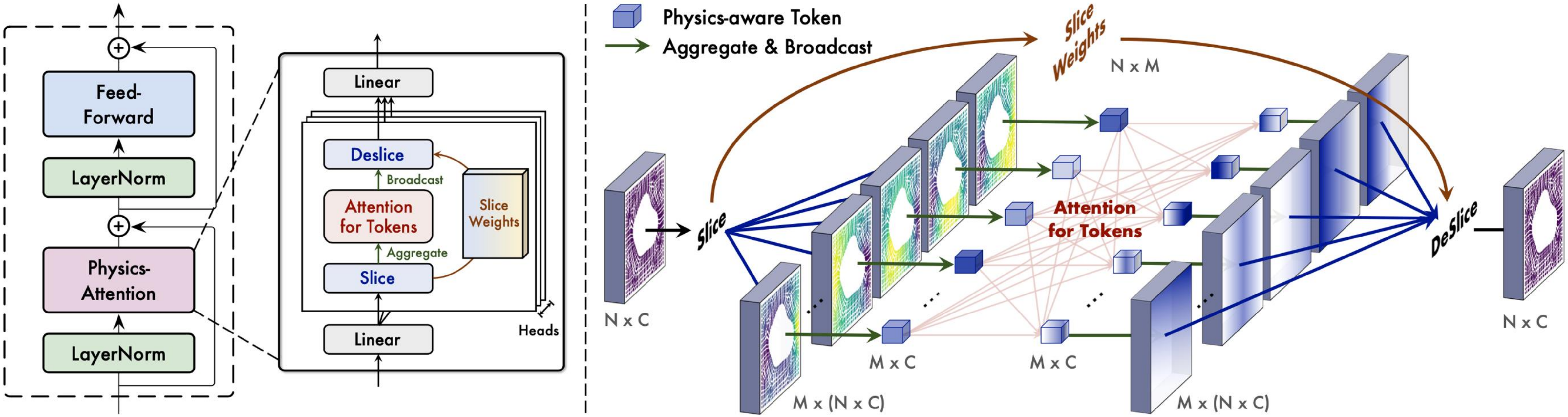
Learning Physical States



(a) Slices for Darcy, 2D Regular Grid
(b) Slices for Elasticity, 2D Point Cloud
(c) Slices for Airfoil, 2D Mesh
(d) Slices for Shape-Net Car Surrounding Velocity, 3D Volumes
(e) Slices for Shape-Net Car Surface Pressure, 3D Mesh

Mesh points under **similar physical states** will be ascribed to the same **slice** and then encoded into a **physics-aware token**.

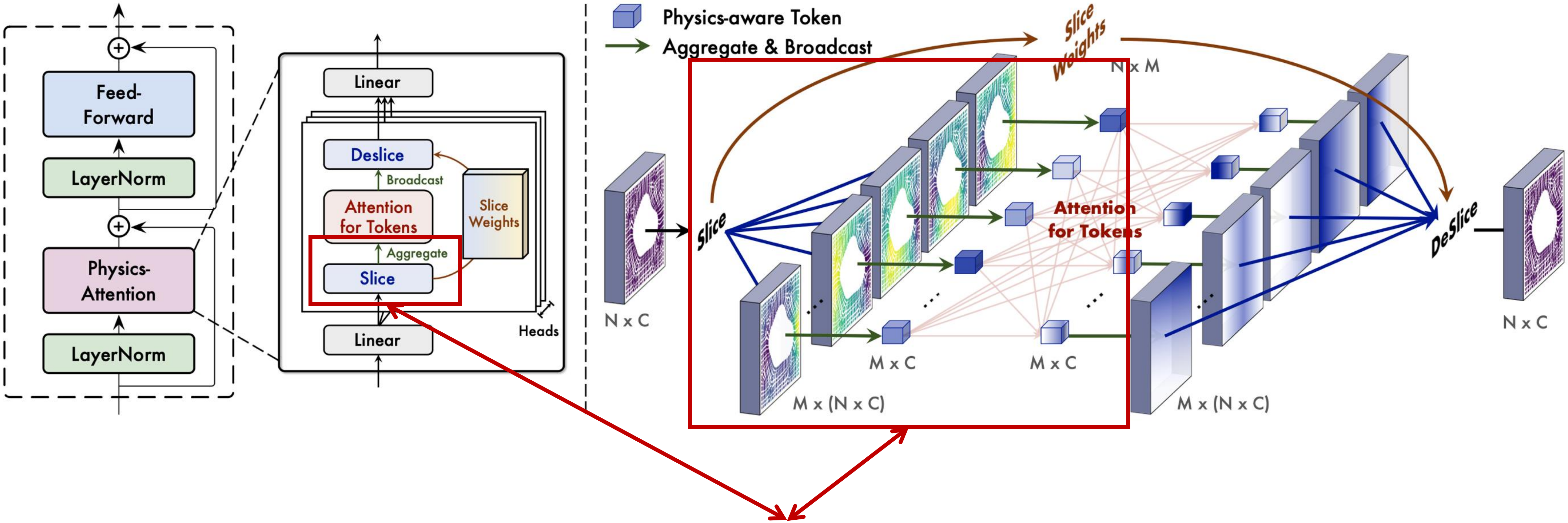
Overview of Transolver



Transolver applies attention to learned physical states **(Physics-Attention)**

① Mesh → physics ② Attention (Integral) ③ Physics → Mesh

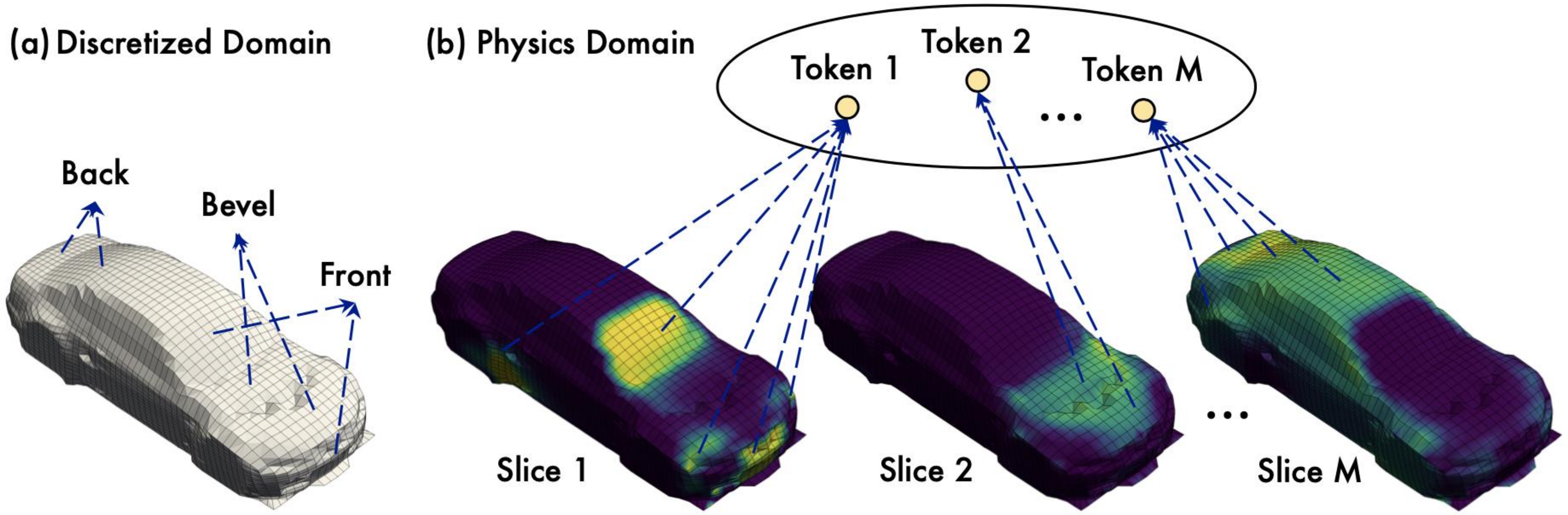
Overview of Transolver



① Mesh → physics

To obtain physics-aware tokens

Mesh → physics



1. Assign each point to slices with weights learned from features

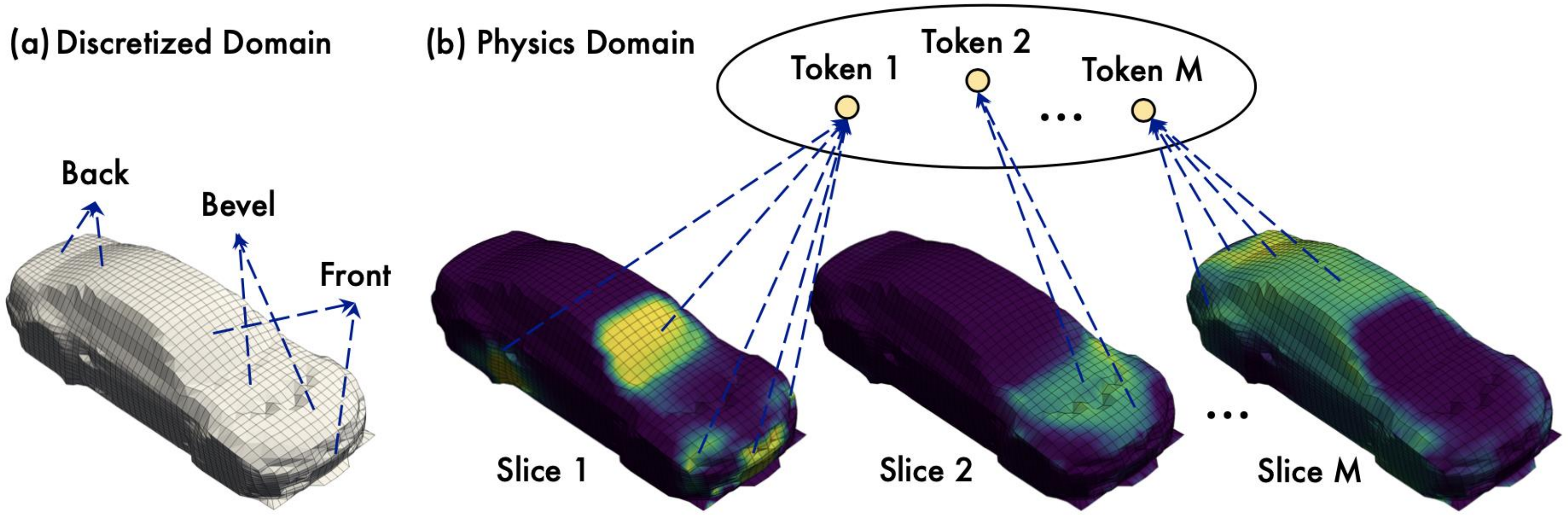
$$\{\mathbf{w}_i\}_{i=1}^{N} = \left\{\underline{\mathrm{Softmax}\,(}\,\mathrm{Project}\,(\mathbf{x}_i)\,)\right\}_{i=1}^{N}$$

$$\mathbf{s}_j = \{\mathbf{w}_{i,j}\mathbf{x}_i\}_{i=1}^{N},$$

***N* Points to *M* Slices**

Softmax for low-entropy slices

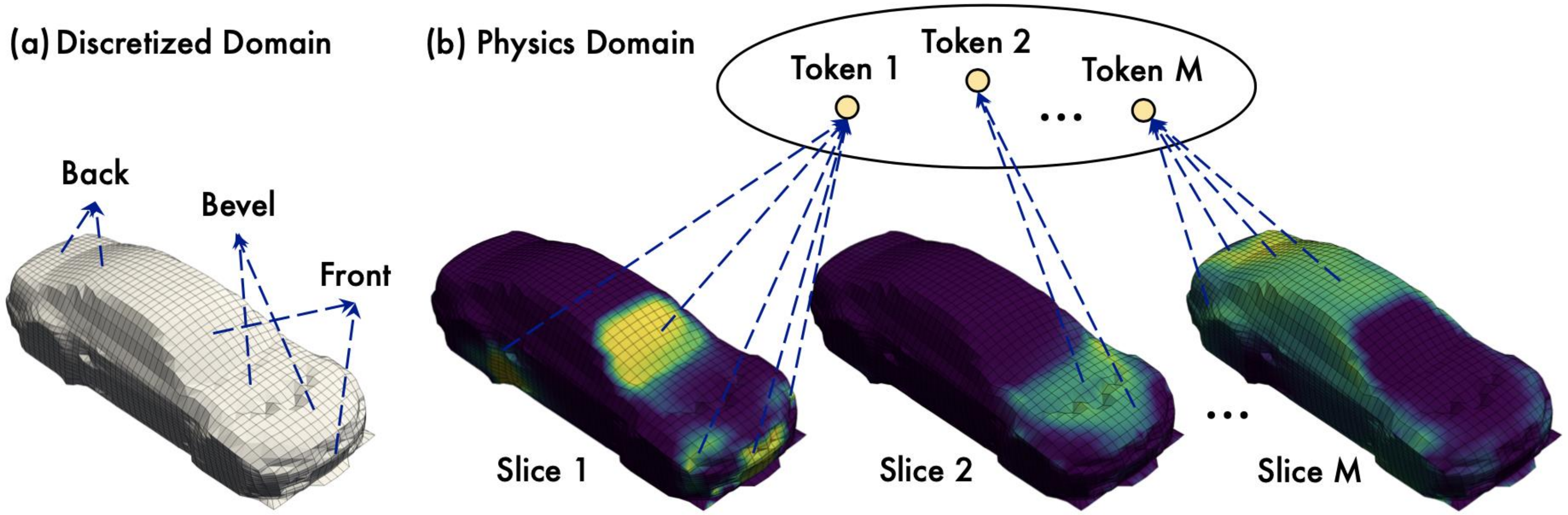
Mesh → physics



1. Assign each point to slices 2. Aggregate slices for physics-aware tokens

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

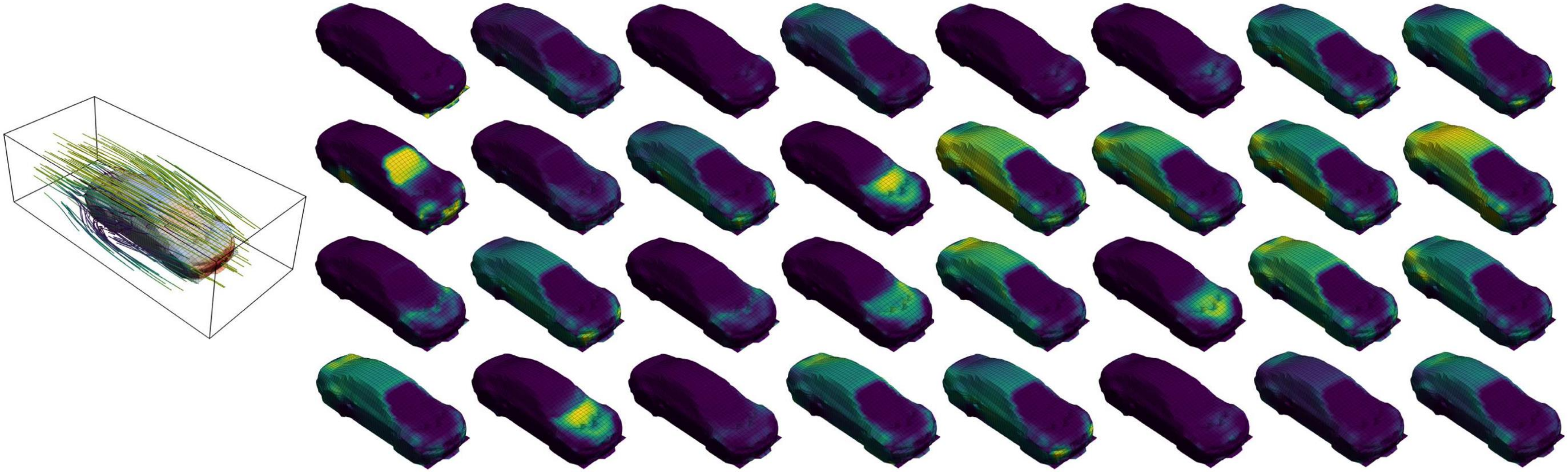
Mesh → physics



1. Why slices can learn physically internal-consistent information?
2. Learning slice is different from splitting computation area

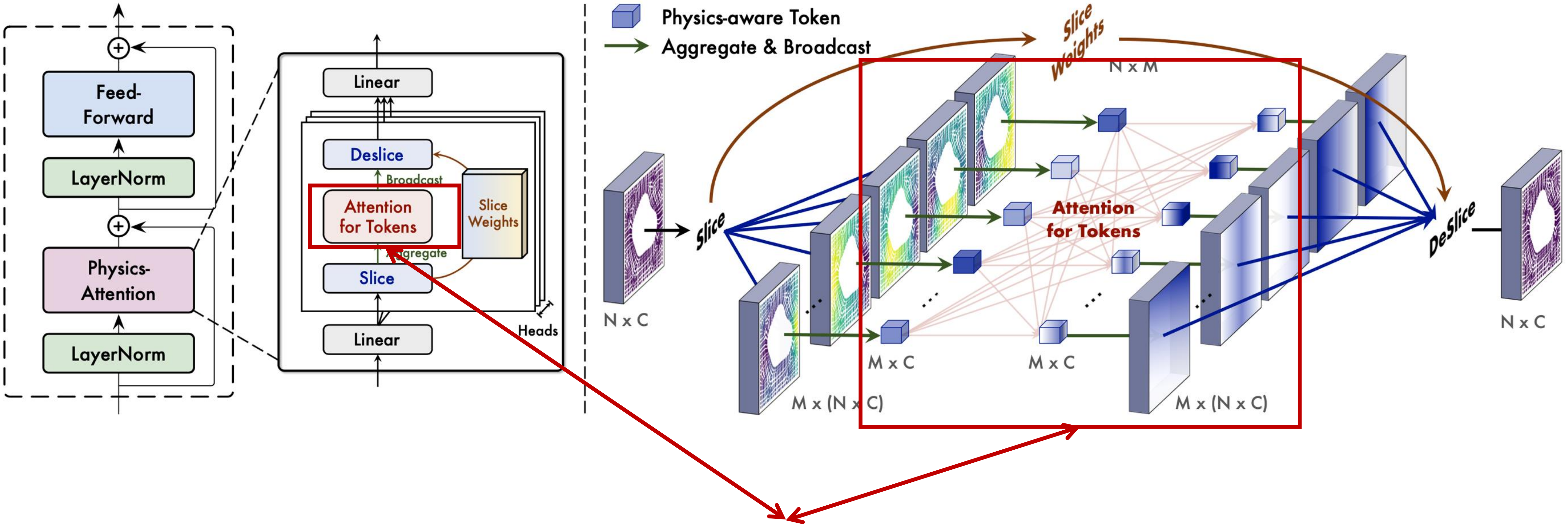
   Ascribe physically similar but spatially distant points to the same slice

Visualization of Learned Slices



M is set as 32. Different slices capture different patterns.

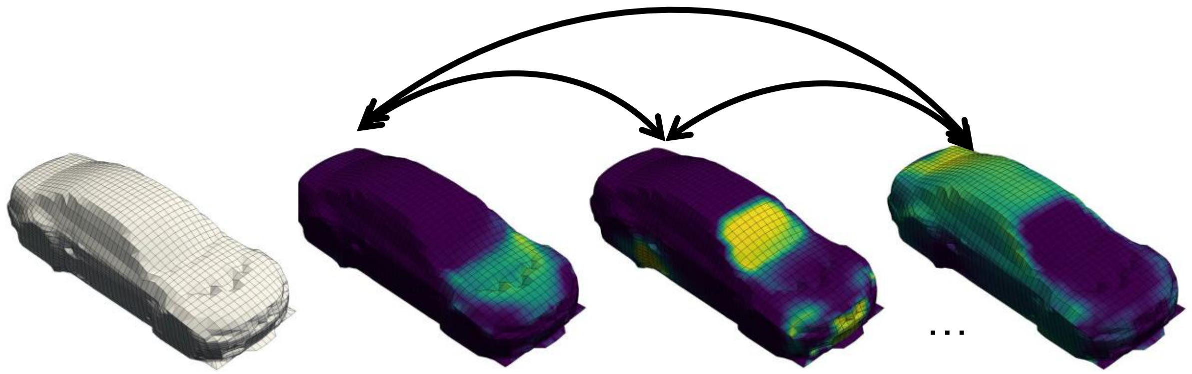
Overview of Transolver



② Attention among physics tokens

Approximate Integral to solve PDEs

Attention among physics tokens

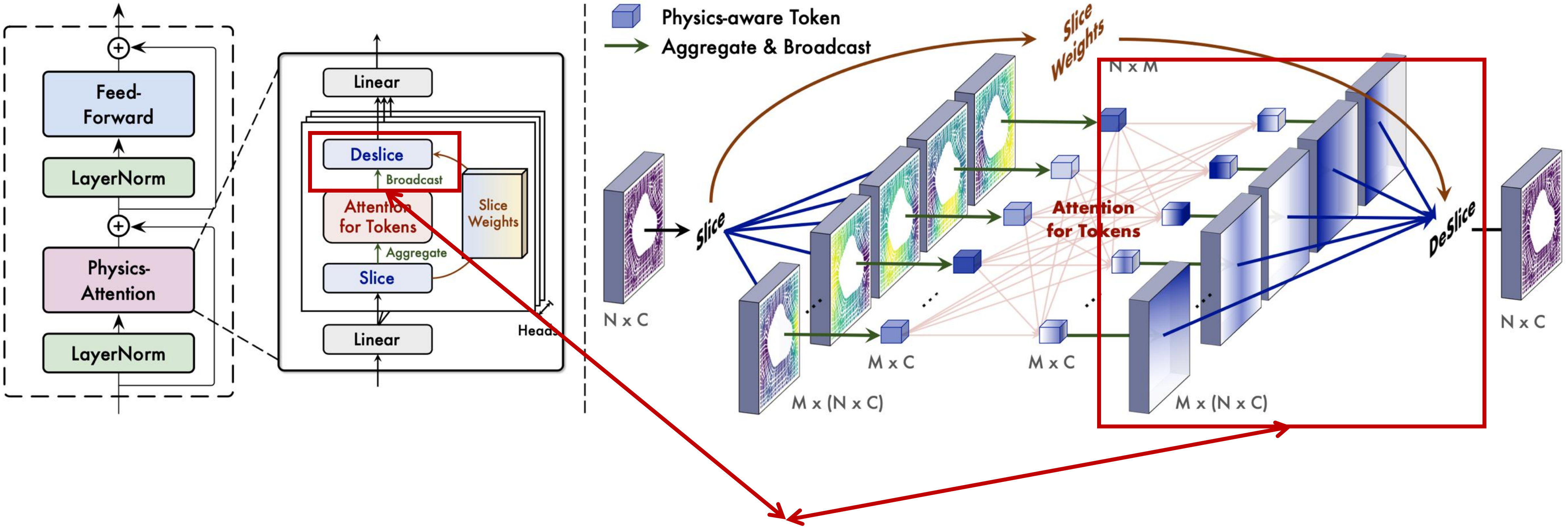


$$\mathbf{q}, \mathbf{k}, \mathbf{v} = \mathrm{Linear}(\underline{\mathbf{z}}), \quad \mathbf{z}' = \mathrm{Softmax}\left(\frac{\mathbf{q}\mathbf{k}^{\mathsf{T}}}{\sqrt{C}}\right)\mathbf{v}$$

Canonical attention among physics tokens

1. Complexity: $\mathcal{O}(N^2C) \to \mathcal{O}(M^2C)$
2. Capture interactions among physics states
3. Theorem: Attention as learnable integral operator

Overview of Transolver



③ Physics → Mesh

Project physics information back to mesh

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

Theoretical Understanding of Transolver

1. Corollary of *Attention is a learnable integral*

Since attention mechanism is applied to tokens encoded from slices, **the step 2 (attention part of Transolver) is a learnable integral for the <u>physics domain</u>**

Is Physics-Attention still an <u>input domain</u> integral?

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

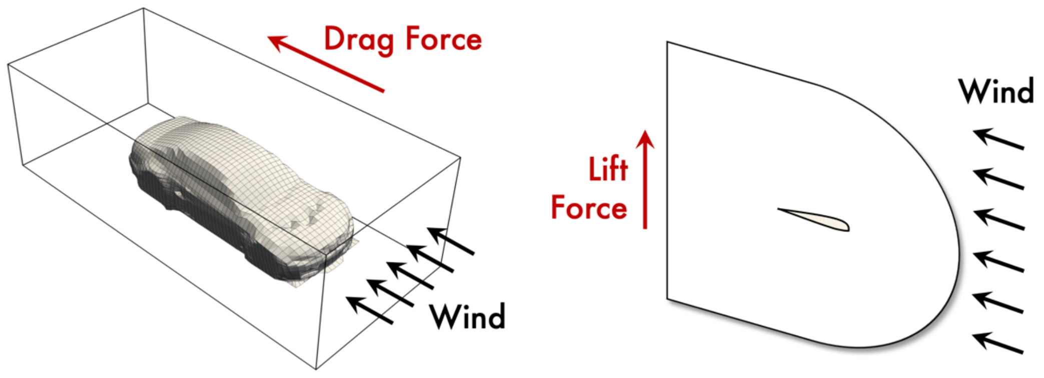
Theoretical Understanding of Transolver

$$
\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}}\left(\mathbf{g}, \boldsymbol{\xi}_{\mathrm{s}}\right)\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}}\left(\mathbf{g}, \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}} \\
&= \int_{\Omega_{\mathrm{s}}} \left(\frac{\int_{\Omega_{\mathrm{s}}} w_{\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}^{\prime}}\kappa_{\mathrm{ss}}\left(\boldsymbol{\xi}_{\mathrm{s}}^{\prime}, \boldsymbol{\xi}_{\mathrm{s}}\right)\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}^{\prime}}{\int_{\Omega_{\mathrm{s}}} w_{\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}^{\prime}}\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}^{\prime}}\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}}^{\prime}}}_{\text{DeSlice}} \int_{\Omega_{\mathrm{s}}} \underbrace{\kappa_{\mathrm{ss}}\left(\boldsymbol{\xi}_{\mathrm{s}}^{\prime}, \boldsymbol{\xi}_{\mathrm{s}}\right)}_{\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}}^{\prime} && \left(\text{Suppose that } \int_{\Omega_{\mathrm{s}}} w_{\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}^{\prime}}\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}^{\prime} = 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}
$$

All the designs in Transolver can be directly derived.

Experiments

GEOMETRY	BENCHMARKS	#DIM	#MESH
POINT CLOUD	ELASTICITY	2D	972
STRUCTURED MESH	PLASTICITY AIRFOIL PIPE	2D+TIME 2D 2D	3,131 11,271 16,641
REGULAR GRID	NAVIER–STOKES DARCY	2D+TIME 2D	4,096 7,225
UNSTRUCTURED MESH	SHAPE-NET CAR AIRFRANS	3D 2D	32,186 32,000



(a) Shape-Net Car (b) AirfRANS

Six standard benchmarks, two practical design tasks

More than 20 baselines

Standard PDE-Solving Benchmarks

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%

Transolver achieves 22% error reduction over the second-best model

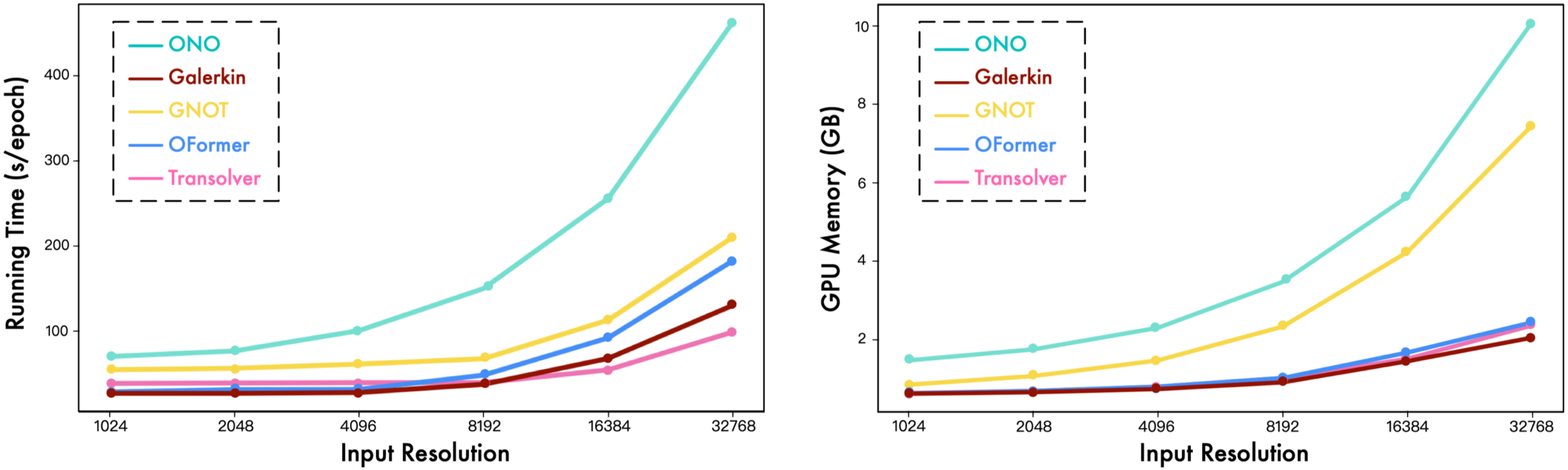
Practical Design Tasks

Model*	Shape-Net Car				AirfRANS			
	Volume ↓	Surf ↓	C_D ↓	ρ_D ↑	Volume ↓	Surf ↓	C_L ↓	ρ_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**

Design-oriented metrics: Drag/lift coefficients and their Spearman's correlation

Transolver performs best in both physics and design-oriented metrics

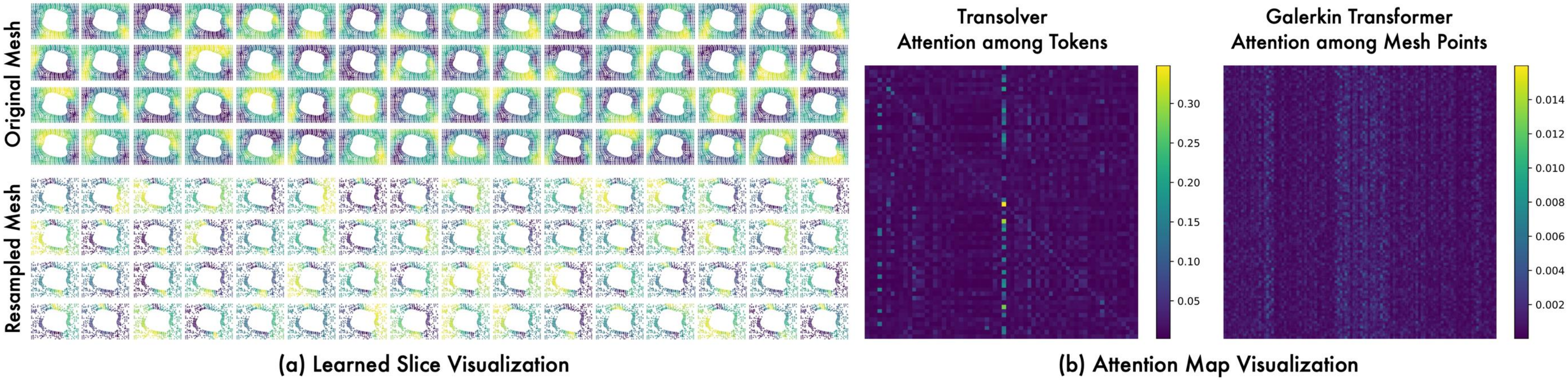
Efficiency



Favorable efficiency and performance balance

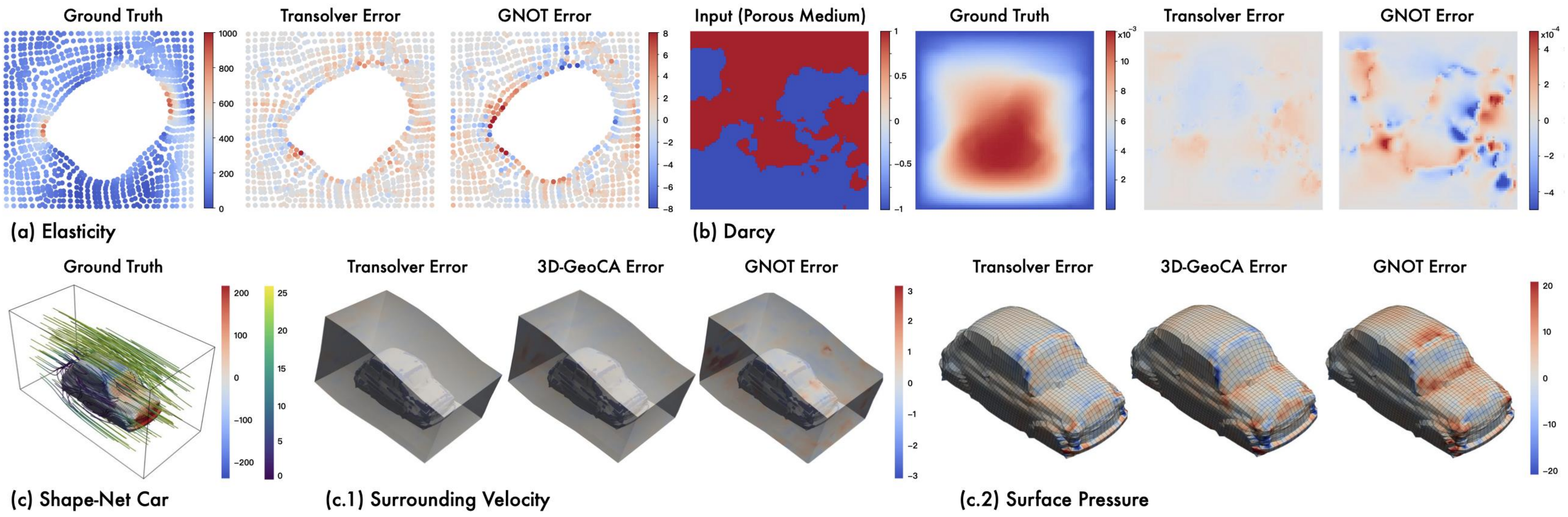
Transolver is faster than linear Transformers in large-scale meshes.

Attention Visualization



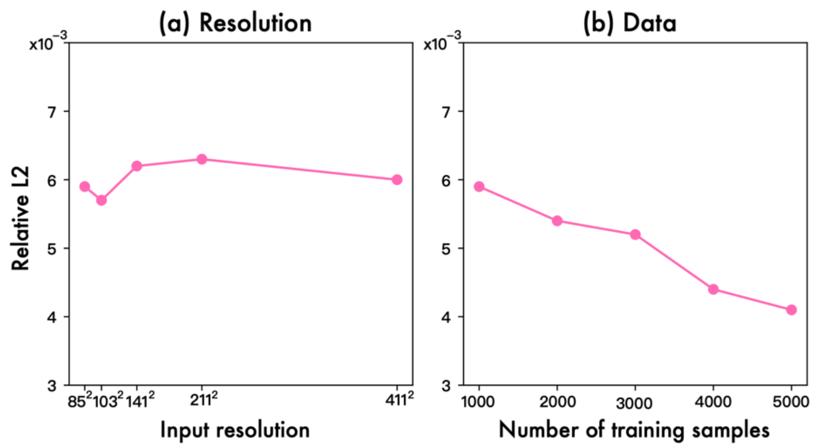
Physics-attention works well even in broken meshes and achieves **more concentrated attention**.

Showcases

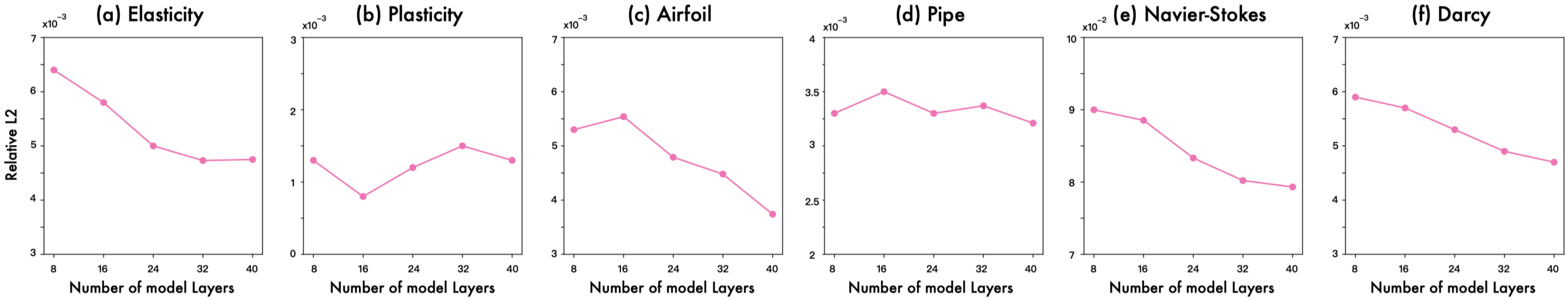


Transolver excels in solving **multiphysics PDEs on hybrid geometrics**

Pursuing PDE Foundation Models: Scalability

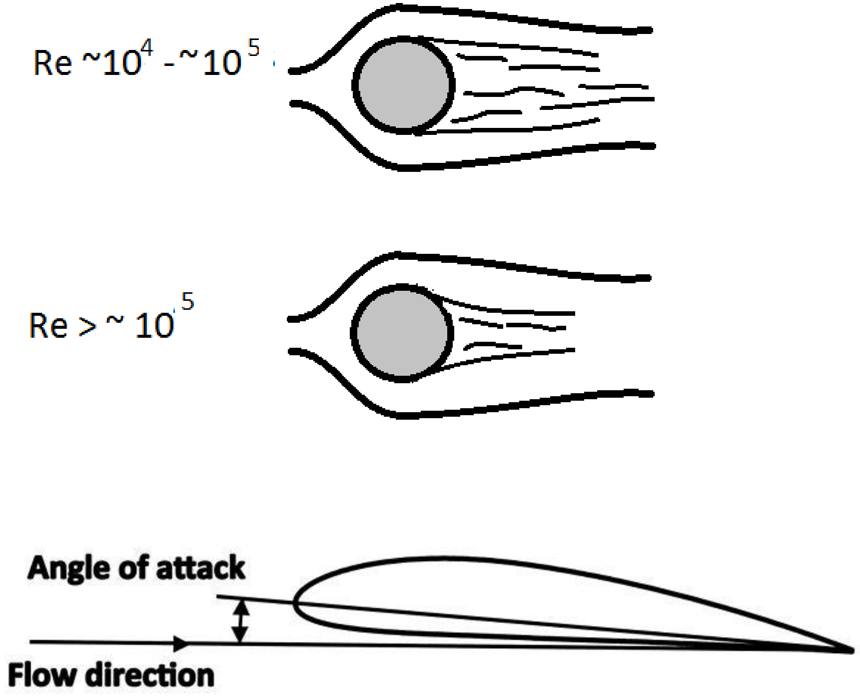


1. **Resolution:** Consistent performance at varied scales
2. **Data:** Benefiting from larger training data
3. **Parameter:** Benefiting from more parameters



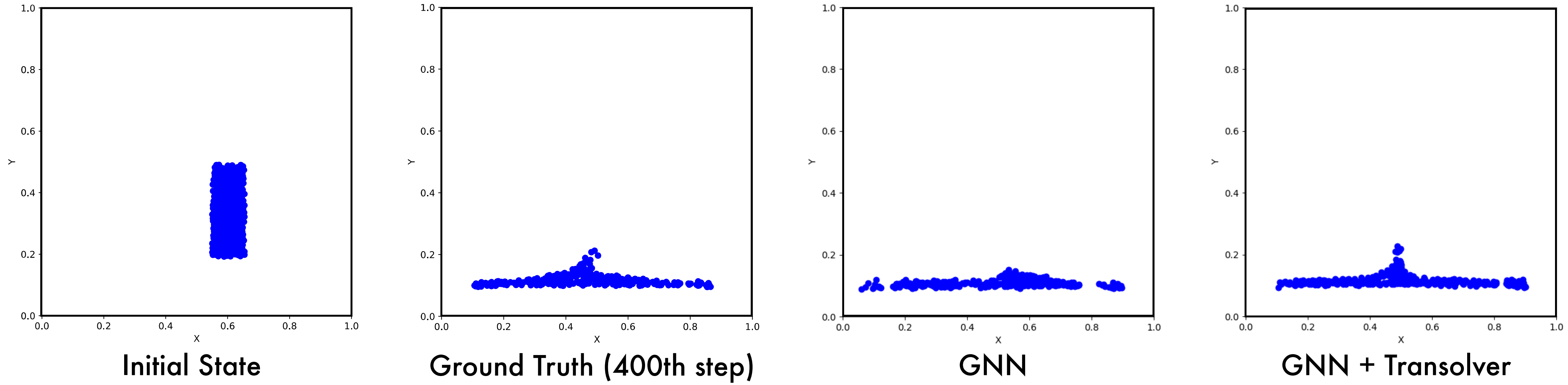
Pursuing PDE Foundation Models: Generalization

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	0.2538	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	0.9878	0.3038	0.9884
Galerkin (2021)	0.4615	0.9826	0.3814	0.9821
GNOT (2023)	0.3268	0.9865	0.3497	0.9868
GINO (2023a)	0.4180	0.9645	0.2583	0.9923
Transolver (Ours)	**0.2996**	**0.9896**	**0.1500**	**0.9950**



Transolver still performs best (**Spearman's correlation ~ 99%**) in OOD settings

Pursuing PDE Foundation Models: Versatile



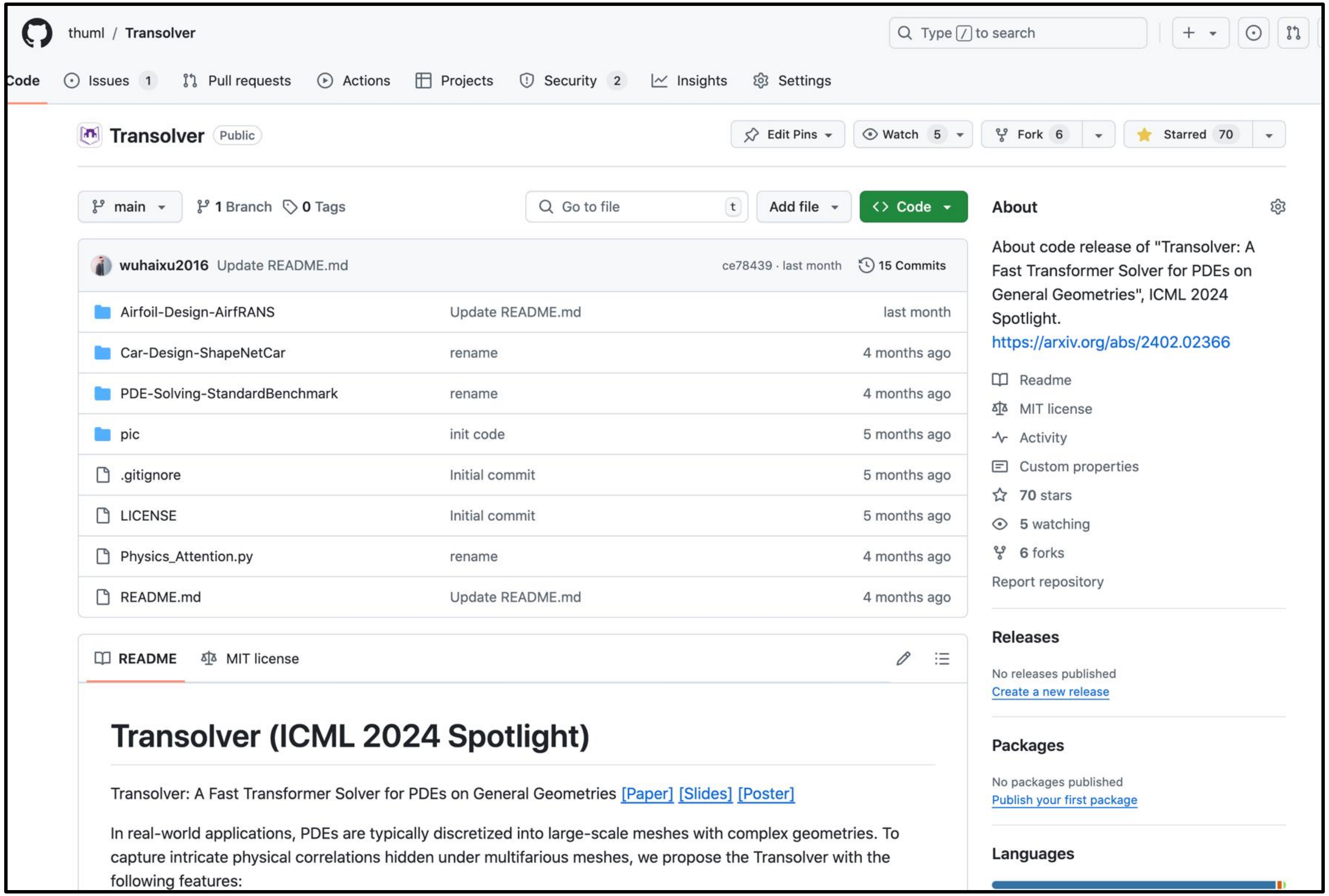
Model	MSE ↓
GNN (Sanchez-Gonzalez et al., 2020)	0.0182
GNN + Transolver (Ours)	**0.0069**
Relative Promotion	62.1%

Transolver can also be extended to

Lagrangian Settings

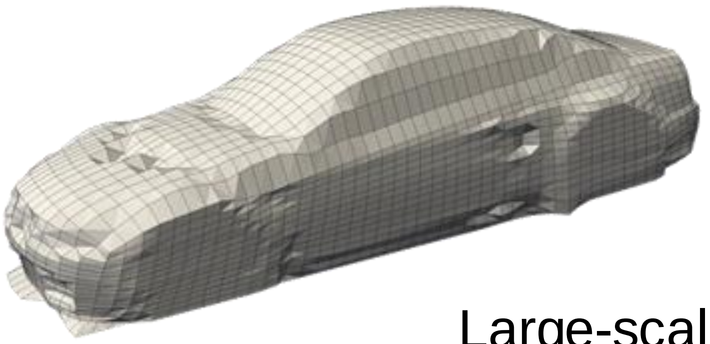
(Ever-changing geometrics)

Open Source

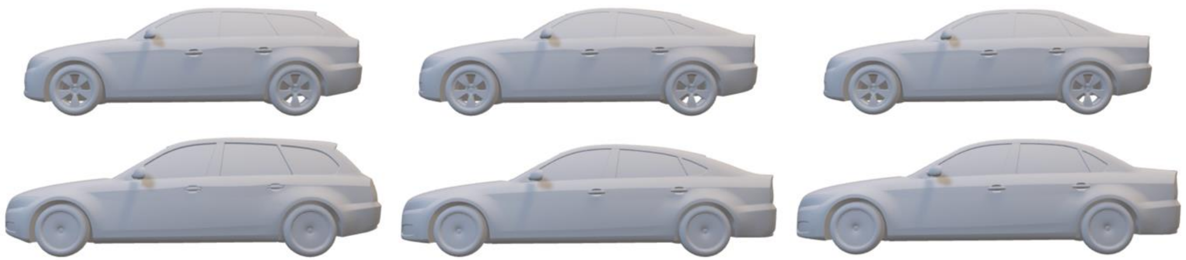


Code is available at https://github.com/thuml/Transolver

Our Exploration for Practical Neural PDE Solvers



Large-scale Meshes



Diverse PDEs, e.g. boundaries, coefficients, forces

1. Foundation Backbone:

Transolver

2. Generalizable Model:

Unisolver

Unisolver: PDE-Conditional Transformers Are Universal PDE Solvers

Hang Zhou[*], Yuezhou Ma[*], Haixu Wu[✉], Haowen Wang, Mingsheng Long[✉]
School of Software, BNRist, Tsinghua University, China

Hang Zhou

Yuezhou Ma

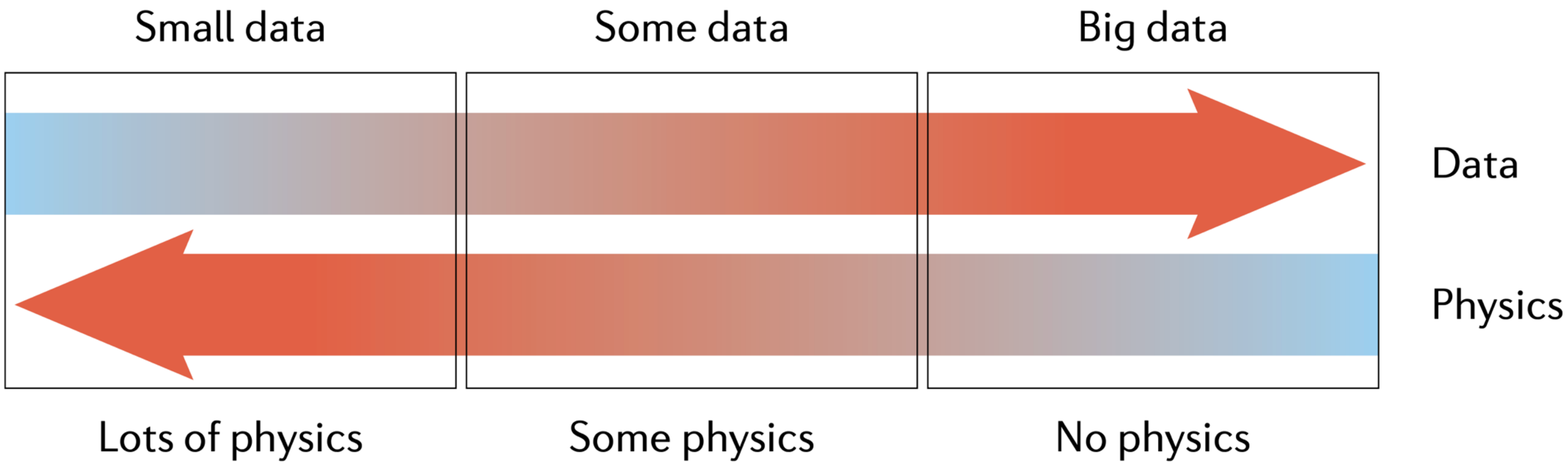
Haixu Wu

Haowen Wang

Mingsheng Long

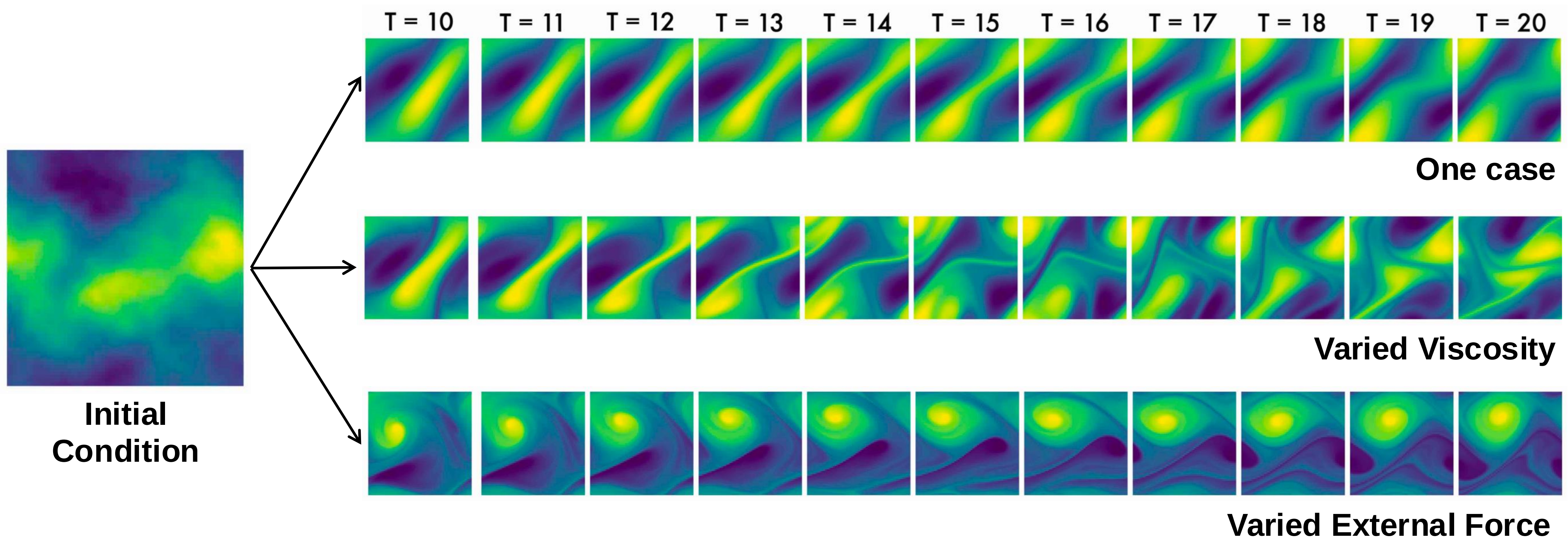
Neural Solvers for PDEs

Physics-Informed Neural Networks ⇔ Neural Operators



Karniadakis, G. et al. Physics-informed machine learning, ***Nature Review Physics*** 3, 422–440 (2021)

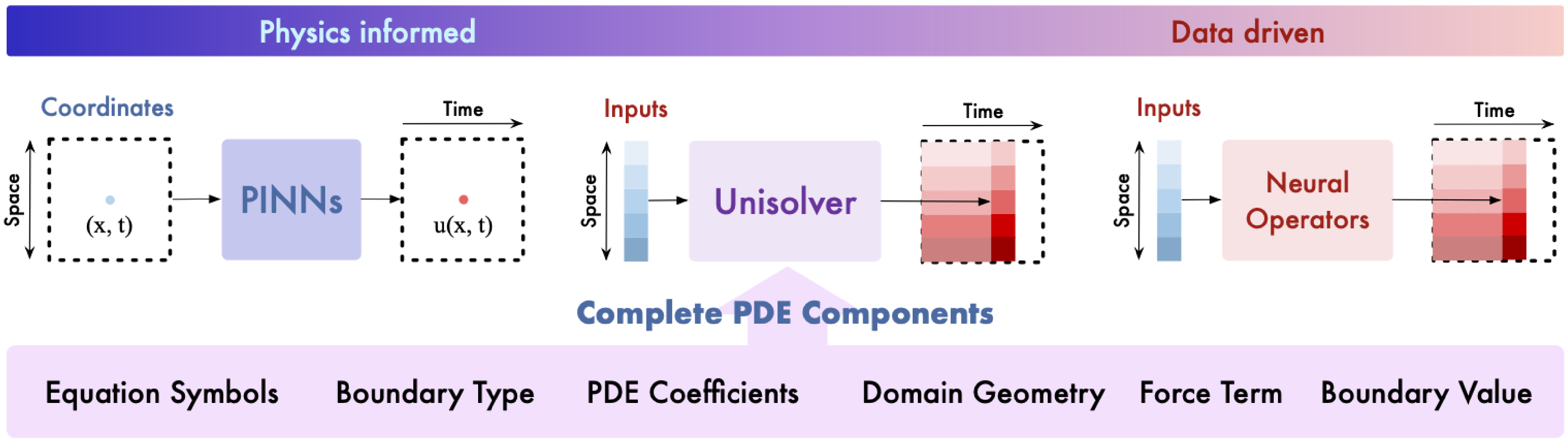
What will Happen without PDE Information?



Previous standard benchmarks only contain one type of viscosity and force.

Beyond data, we also need to know what kind of PDE we attempt to solve.

Unisolver: A Unification of Two Paradigms



In addition to simulated data, Unisolver also defines and utilizes

a complete set of PDE components.

Complete PDE Components

Motivating example: vibrating string equation

$$
\begin{aligned}
&\partial_{tt}u - a^2\partial_{xx}u = f(x,t), && (x,t) \in (0,L) \times (0,T), && \text{(1a)}\\
&u(0,t) = 0,\ u(L,t) = 0, && t \in (0,T], && \text{(1b)}\\
&u(x,0) = \phi(x),\ \partial_t u(x,0) = \psi(x), && x \in [0,L]. && \text{(1c)}
\end{aligned}
$$

- The coefficient a represents physical quantity such as tension, linear density
- f represents the external force driving the vibrations of the string
- Equation (1b) sets boundary conditions at endpoints
- Equation (1c) specifies initial conditions
- The domain geometry spans the range $[0, L] \times [0, T]$

Complete PDE Components

Motivating example: vibrating string equation

$$\begin{aligned}
&\partial_{tt}u - a^2\partial_{xx}u = f(x,t), && (x,t) \in (0,L)\times(0,T), && \text{(1a)}\\
&u(0,t) = 0,\ u(L,t) = 0, && t \in (0,T], && \text{(1b)}\\
&u(x,0) = \phi(x),\ \partial_t u(x,0) = \psi(x), && x \in [0,L]. && \text{(1c)}
\end{aligned}$$

- The analytical solution of the above equations is:

$$u(x,t) = \frac{1}{2}\underbrace{(\Phi(x+at)+\Phi(x-at))}_{\textit{Initial position}} + \frac{1}{2a}\underbrace{\int_{x-at}^{x+at}\Psi(\xi)\,\mathrm{d}\xi}_{\textit{Initial velocity}} + \frac{1}{2a}\underbrace{\int_0^t \mathrm{d}\tau \int_{x-a(t-\tau)}^{x+a(t-\tau)}}_{\textit{Geometry}} \underbrace{f(\xi,\tau)}_{\textit{Force}}\,\mathrm{d}\xi$$

- The PDE is solved under complex interactions between equation components
- The impact of the external force is imposed point-wisely
- The coefficient exerts a consistent influence over the domain

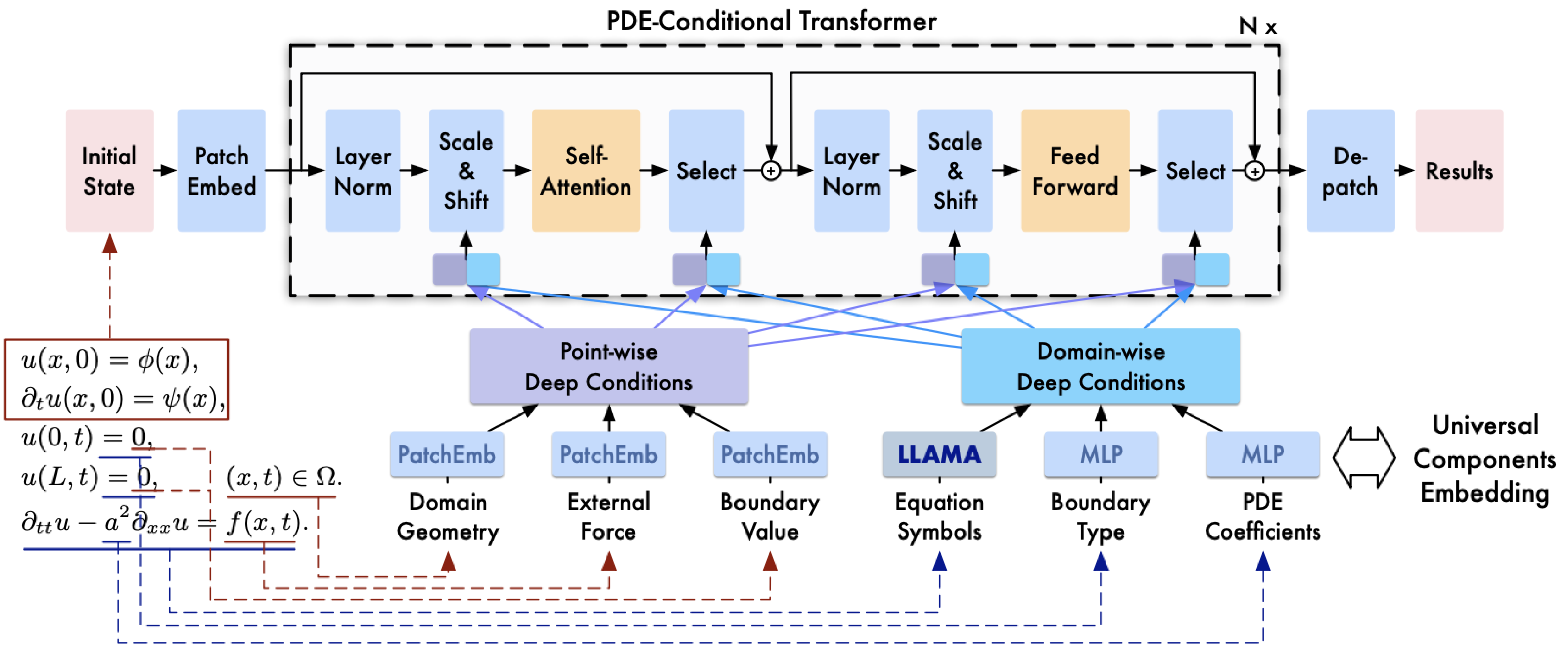
Complete PDE Components

Category	Component	Description
Domain-wise components	Equation formulation	The symbolic expression of the PDE
	Equation coefficient	Coefficients in the PDE equation
	Boundary condition type	Type of boundary condition (e.g., Dirichlet, Neumann)
Point-wise components	External force	Forces acting at specific points
	Domain geometry	The shape and size of the domain
	Boundary value function	Value functions at the domain boundaries

- Category PDE components into domain- and point-wise components:
- Here the equation formulation refers to the **symbolic expression of PDEs**, which can be encoded by **Large Language Models**

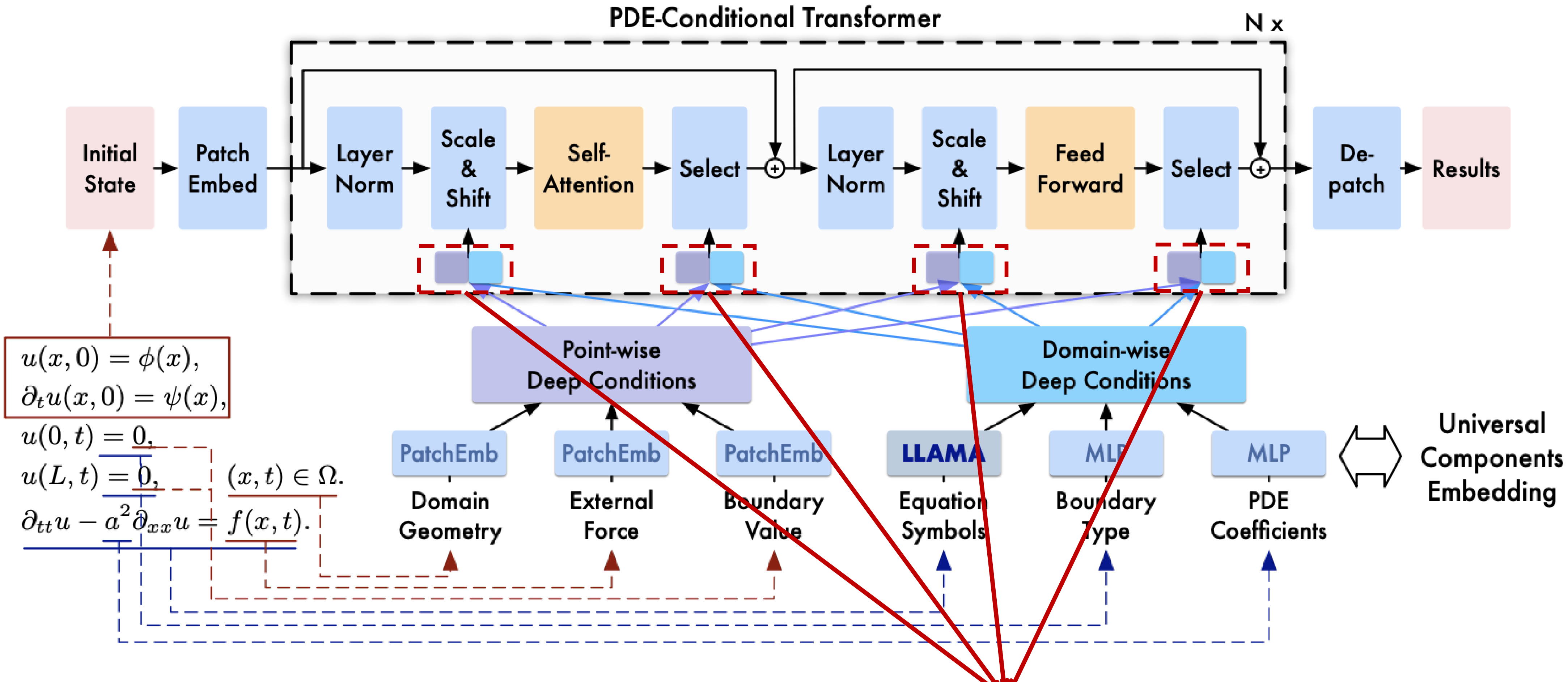
```
Prompt: "u_{tt} - a^2 u_{xx} = f(x,t)"
```

PDE-Conditional Transformer



① Unify Embedding ② Condition Aggregation

PDE-Conditional Transformer

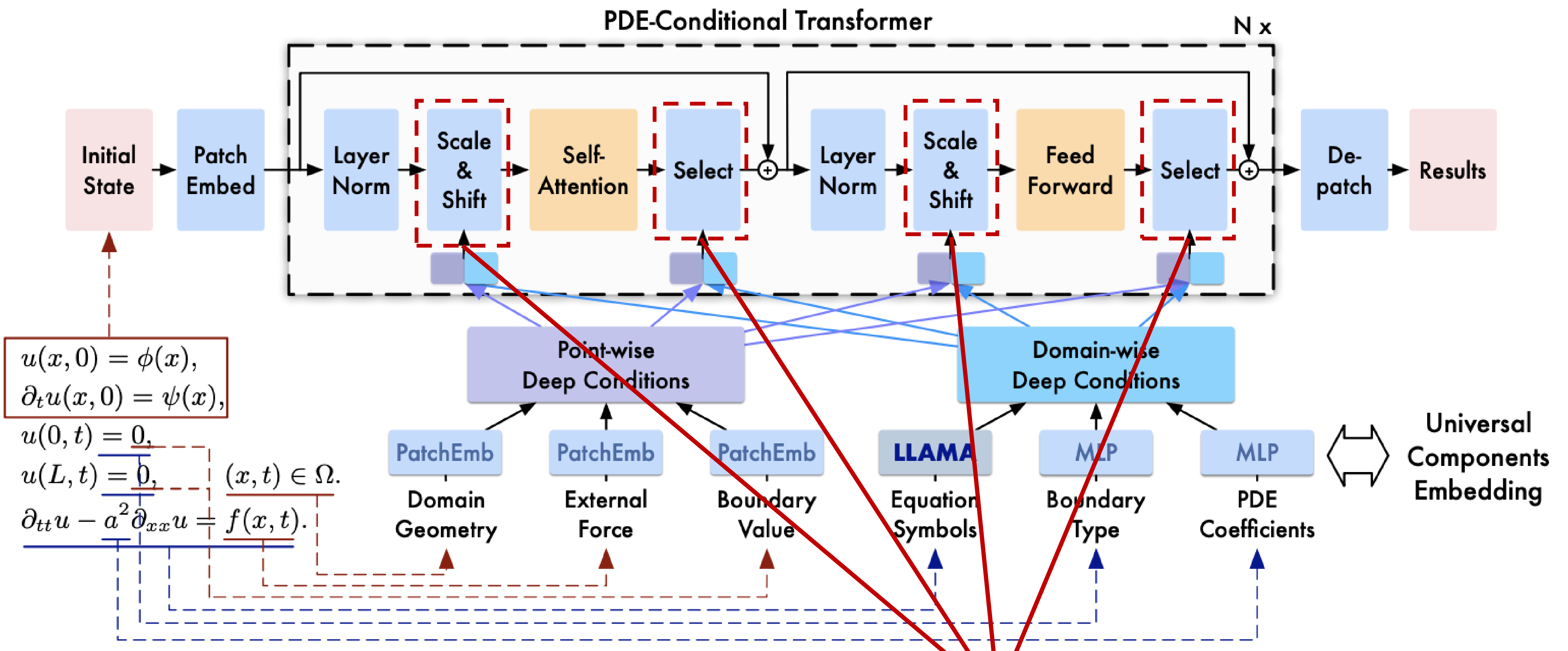


① Unify Embedding

$$\mathbf{I}_* = \mathrm{Concat}\left(\mathrm{MLP}_*(\mathbf{C}_{\mathrm{domain}}).\mathtt{repeat}, \mathrm{MLP}_*(\mathbf{C}_{\mathrm{point}})\right), \quad * \in \{\text{scale, shift, select}\}$$

$$\hat{\mathbf{I}}_* = \mathrm{Concat}\left(\widehat{\mathrm{MLP}}_*(\mathbf{C}_{\mathrm{domain}}).\mathtt{repeat}, \widehat{\mathrm{MLP}}_*(\mathbf{C}_{\mathrm{point}})\right), \quad * \in \{\text{scale, shift, select}\}$$

PDE-Conditional Transformer



② Condition Aggregation

$$\widehat{\mathbf{X}}^{n-1} = \mathbf{I}_{\text{select}} \odot \text{SelfAttention}\left(\mathbf{I}_{\text{scale}} \odot \text{LayerNorm}(\mathbf{X}^{n-1}) + \mathbf{I}_{\text{shift}}\right) + \mathbf{X}^{n-1},$$

$$\mathbf{X}^{n} = \widehat{\mathbf{I}}_{\text{select}} \odot \text{FeedForward}\left(\widehat{\mathbf{I}}_{\text{scale}} \odot \text{LayerNorm}(\widehat{\mathbf{X}}^{n-1}) + \widehat{\mathbf{I}}_{\text{shift}}\right) + \widehat{\mathbf{X}}^{n-1},$$

Experiments

Benchmarks	#Dim	#Resolution	#Samples	#GPU hours	Symbols	Coefficient	Force	Geometry	Boundary
HeterNS	2D+Time	(64,64,10)	15k	~60h	×	✓	✓	×	×
1D time-dependent PDEs	1D+Time	(256,100)	3M	~3000h	✓	✓	✓	×	✓
2D mixed PDEs	2D+Time	(128,128,10)	74.1k	~800h	✓	✓	✓	✓	✓

- HeterNS contains multiple viscosity coefficients and external force
- PDEformer proposes a large-scale dataset with 3M samples of 1D PDEs, including multiple equation coefficients, external force and boundary conditions
- DPOT collects 12 datasets from FNO, PDEBench, PDEArena and CFDBench, with PDEs varying in coefficients, external force, geometries and boundary conditions

Heterogeneous 2D Navier-Stokes Equation

- In-distribution Test (average **27.4%** promotion over the second best)
- Zero-Shot Generalization (average **43.9%** promotion over the second best)

HeterNS (Viscosity / Params)		In-distribution Test					Zero-shot Generalization					
	Params	ν = 1e-5	ν = 5e-5	ν = 1e-4	ν = 5e-4	ν = 1e-3	ν = 8e-6	ν = 3e-5	ν = 8e-5	ν = 3e-4	ν = 8e-4	ν = 2e-3
FNO	4.7M	0.0669	0.0225	0.0114	0.0031	0.0011	0.0702	0.0373	0.0141	0.0088	0.0084	0.2057
PINO	4.7M	0.1012	0.0443	0.0263	0.0073	0.0031	0.1014	0.0646	0.0299	0.0142	**0.0081**	0.1894
ViT	4.8M	0.0432	0.0206	0.0098	0.0031	0.0015	0.0458	0.0353	0.0119	0.0100	0.0174	0.1878
Factformer	5.1M	0.0571	0.0259	0.0148	0.0018	0.0010	0.0489	0.0642	0.0167	0.1808	0.0639	0.3224
ICON	4.5M	0.0585	0.0267	0.0144	0.0054	0.0029	0.0606	0.0387	0.0169	0.0246	0.0110	0.2149
MPP	4.9M	0.0775	0.0496	0.0321	0.0098	0.0043	0.0796	0.0648	0.0376	0.0387	0.0236	0.2595
Unisolver	4.1M	**0.0321**	**0.0094**	**0.0051**	**0.0015**	**0.0008**	**0.0336**	**0.0178**	**0.0064**	**0.0066**	0.0096	**0.1504**
Promotion	/	25.7%	54.4%	48.0%	16.7%	20.0%	26.6%	49.6%	46.2%	25.0%	/	19.9%

Varied viscosity, Fixed external force

Heterogeneous 2D Navier-Stokes Equation

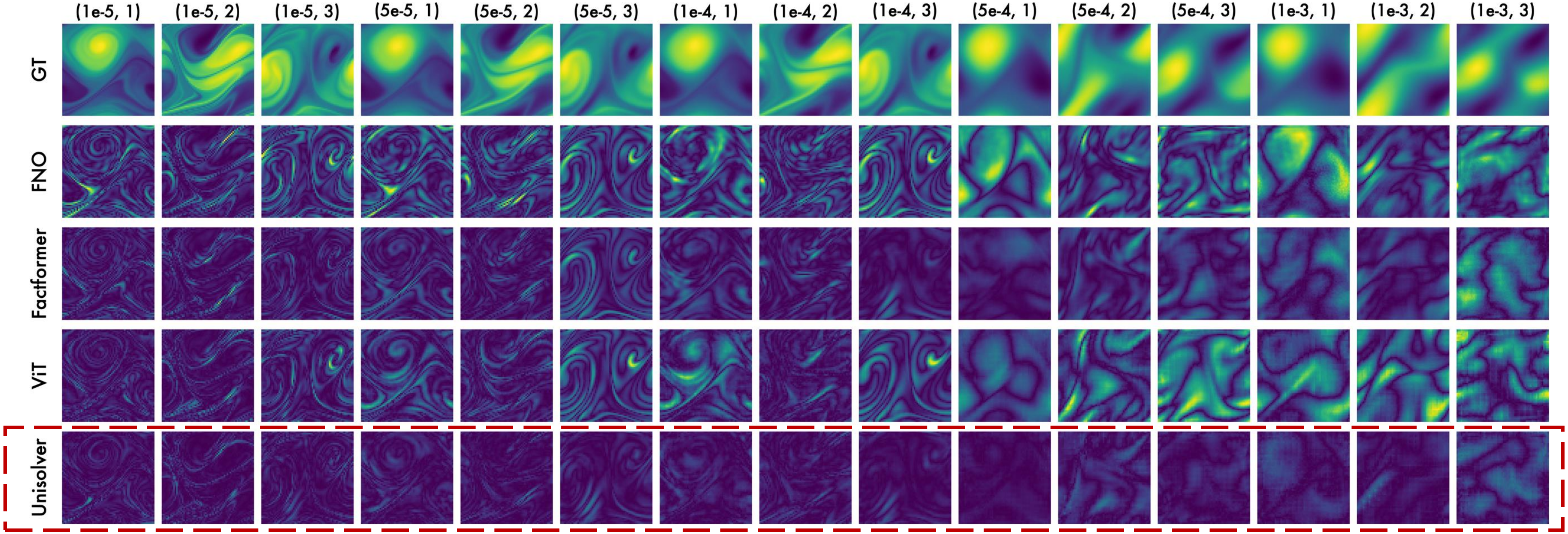
- In-distribution Test (average **27.4%** promotion over the second best)
- Zero-Shot Generalization (average **43.9%** promotion over the second best)

HeterNS	Force / Params	In-distribution Test			Zero-shot Generalization			
		$\omega = 1$	$\omega = 2$	$\omega = 3$	$\omega = 0.5$	$\omega = 1.5$	$\omega = 2.5$	$\omega = 3.5$
FNO	4.7M	0.0640	0.0661	0.1623	1.1100	0.1742	0.1449	0.2974
PINO	4.7M	0.0914	0.1012	0.2707	1.0570	0.5010	0.4660	0.8380
ViT	4.8M	<u>0.0348</u>	<u>0.0432</u>	0.1000	0.7900	0.1412	<u>0.1240</u>	0.2080
Factformer	5.1M	0.0409	0.0570	<u>0.0982</u>	0.8591	<u>0.1207</u>	0.1243	<u>0.2047</u>
ICON	4.5M	0.0435	0.0585	0.1345	1.1950	0.5295	0.5009	0.8231
MPP	4.9M	0.0596	0.0775	0.1620	<u>0.5532</u>	0.2224	0.2180	0.3803
Unisolver	4.1M	**0.0244**	**0.0321**	**0.0720**	**0.0980**	**0.0770**	**0.0720**	**0.1740**
Promotion	/	29.9%	25.7%	26.7%	82.3%	36.2%	41.9%	15.0%

Fixed viscosity, Varied external force

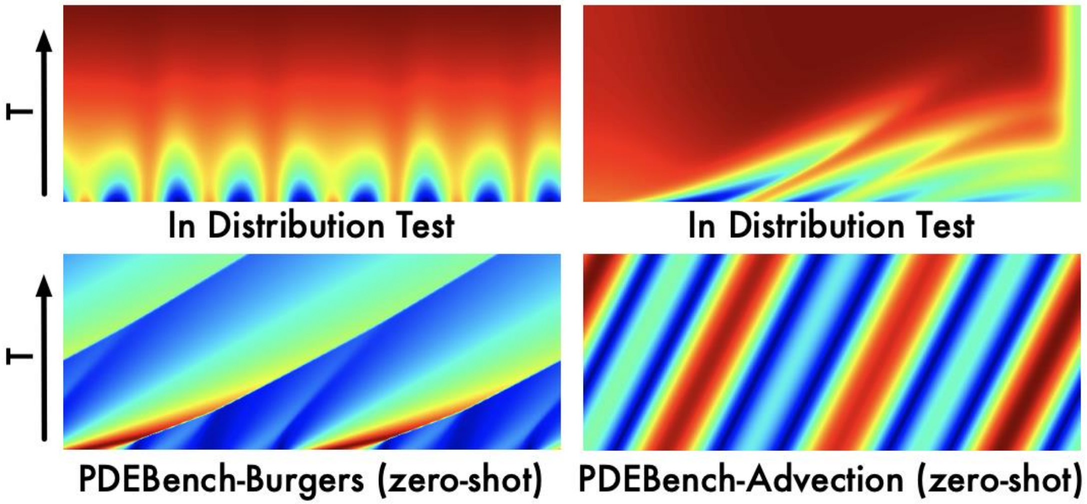
Heterogeneous 2D Navier-Stokes Equation

- All showcases generated with the same initial condition but with varied coefficients. **Different viscosities presents quite different dynamics.**



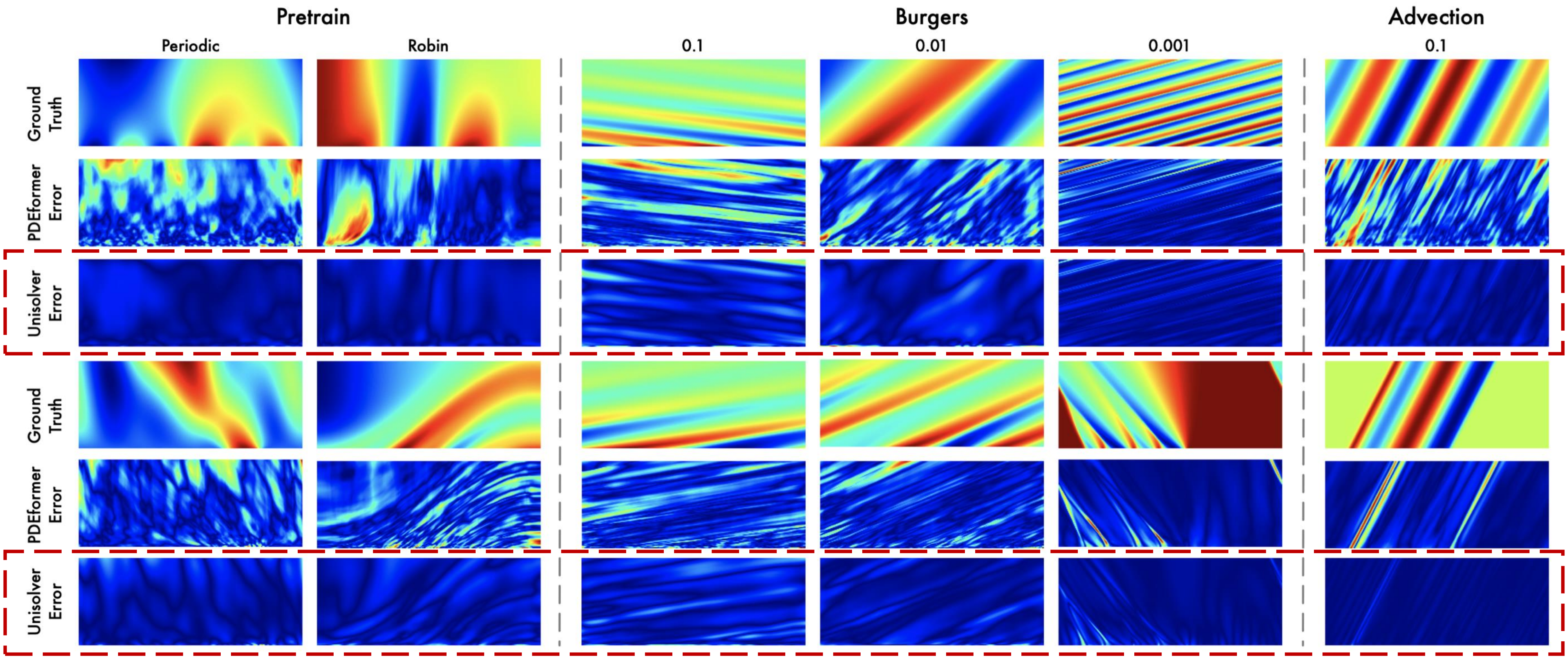
1D Time-dependent PDEs proposed by PDEformer

- **3 million 1D PDEs** with varied **coefficients, external force, boundary conditions and <u>equation symbols</u>**
- **OOD downstream PDE datasets** selected from PDEBench



1D Time-dependent PDEs	Tasks / Params	In-distribution Test	Zero-shot Burgers $\nu = 0.1$	Zero-shot Burgers $\nu = 0.01$	Zero-shot Burgers $\nu = 0.001$	Zero-shot Advection $\beta = 0.1$
PDEformer	22M	0.0225	0.00744	0.0144	0.0393	0.0178
Unisolver	19M	**0.0108**	**0.00513**	**0.00995**	**0.0299**	**0.0138**
Promotion	/	52.0%	31.0%	30.9%	23.9%	22.5%

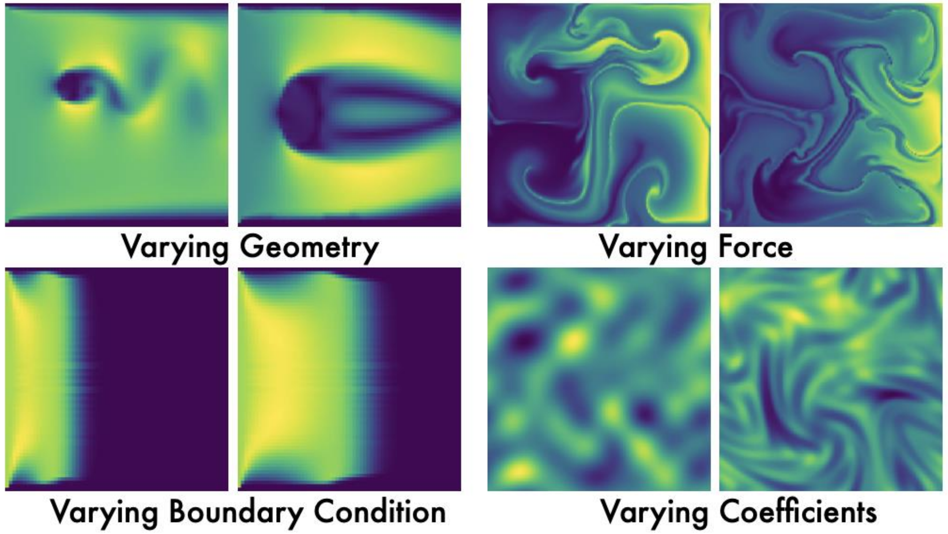
Showcases



Pretrain
Periodic
Robin
Burgers
0.1
0.01
0.001
Advection
0.1
Ground Truth
PDEformer Error
Unisolver Error
Ground Truth
PDEformer Error
Unisolver Error

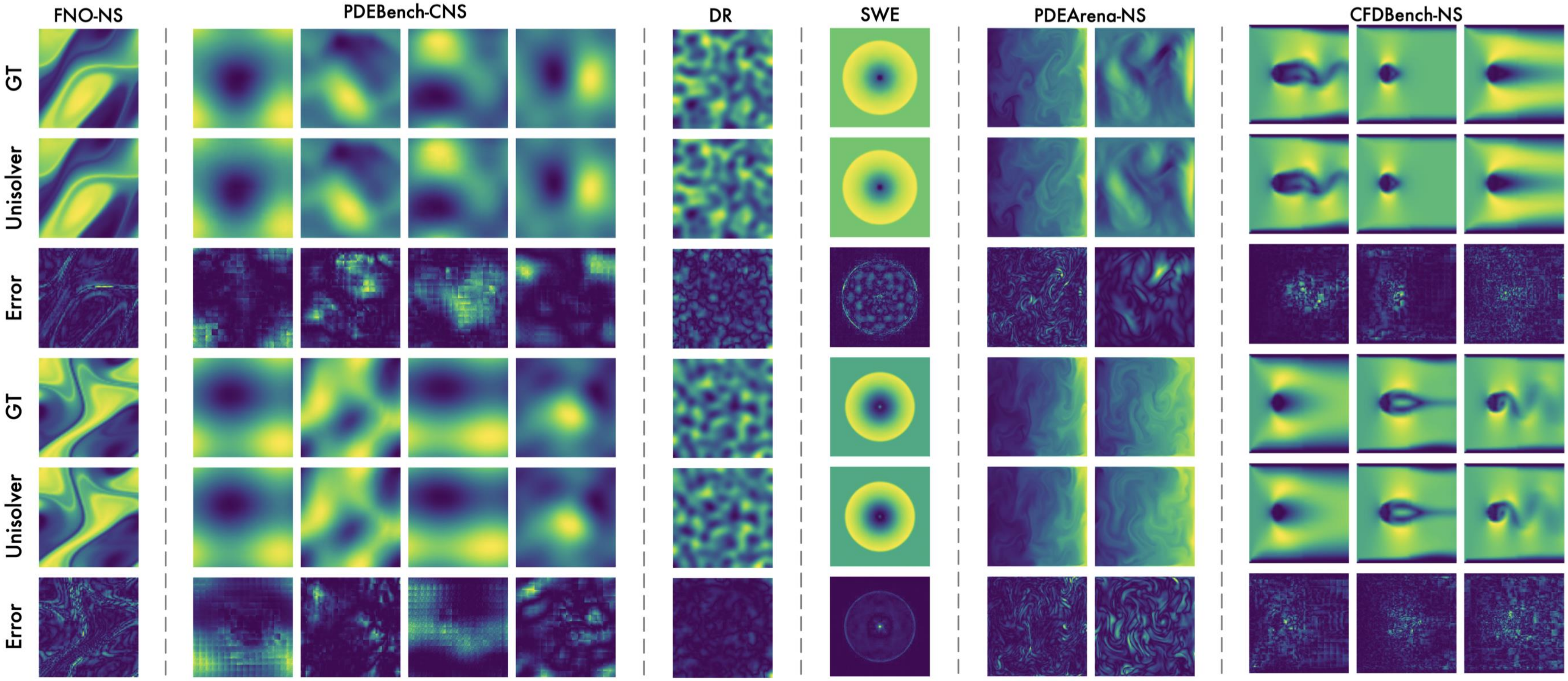
2D Mixed PDEs proposed by DPOT

A dataset mixed from 12 subsets collected by DPOT, with **varied coefficients, external force, boundary conditions and geometries**



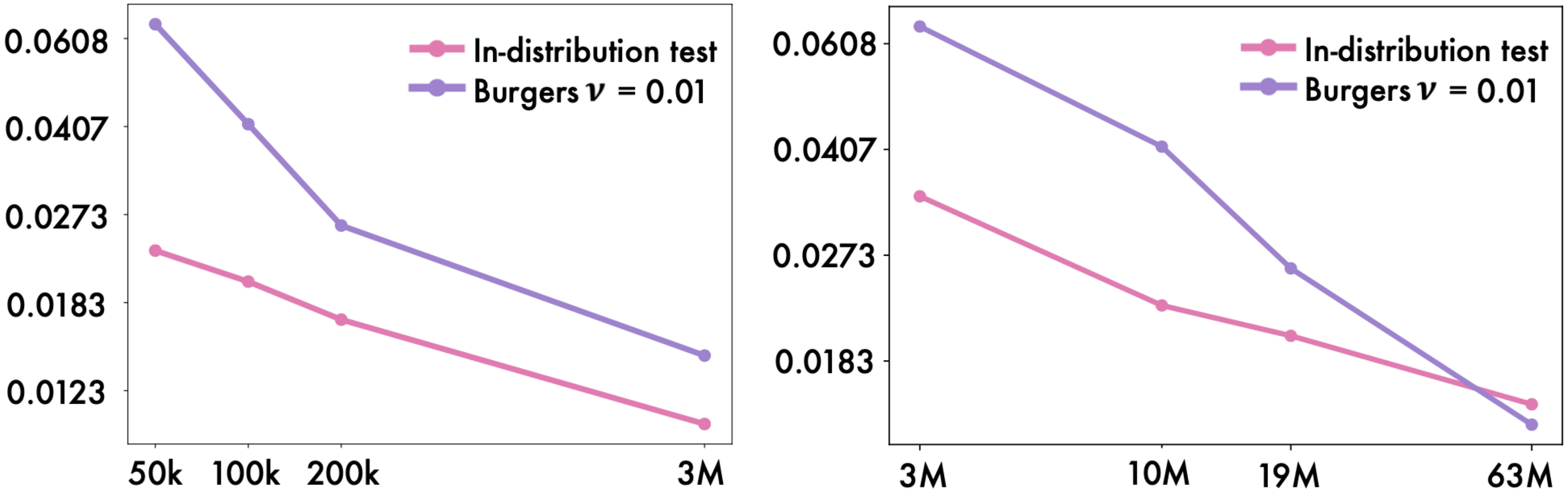
2D Mixed PDEs	Tasks / Params	FNO-NS-ν			PDEBench-CNS-(M,ζ)				PDEBench		PDEArena		CFDBench-NS	Average Error
		1e-5	1e-4	1e-3	(1, 0.1)	(1, 0.01)	(0.1, 0.1)	(0.1, 0.01)	DR	SWE	NS	NS-Force	Geometry	
DPOT	30M	5.53	4.42	1.31	1.53	3.37	1.19	1.87	**3.79**	0.66	9.91	31.6	0.70	5.50
Unisolver	33M	**4.17**	**3.36**	**0.61**	**1.23**	**2.89**	**1.01**	**1.59**	4.39	**0.45**	**6.87**	**27.4**	**0.54**	**4.54**
Promotion (%)	/	24.6	24.0	53.4	19.6	14.2	15.1	15.0	/	31.8	30.7	13.3	22.9	17.5

Showcases



FNO-NS
PDEBench-CNS
DR
SWE
PDEArena-NS
CFDBench-NS
GT
Unisolver
Error
GT
Unisolver
Error

Scalability

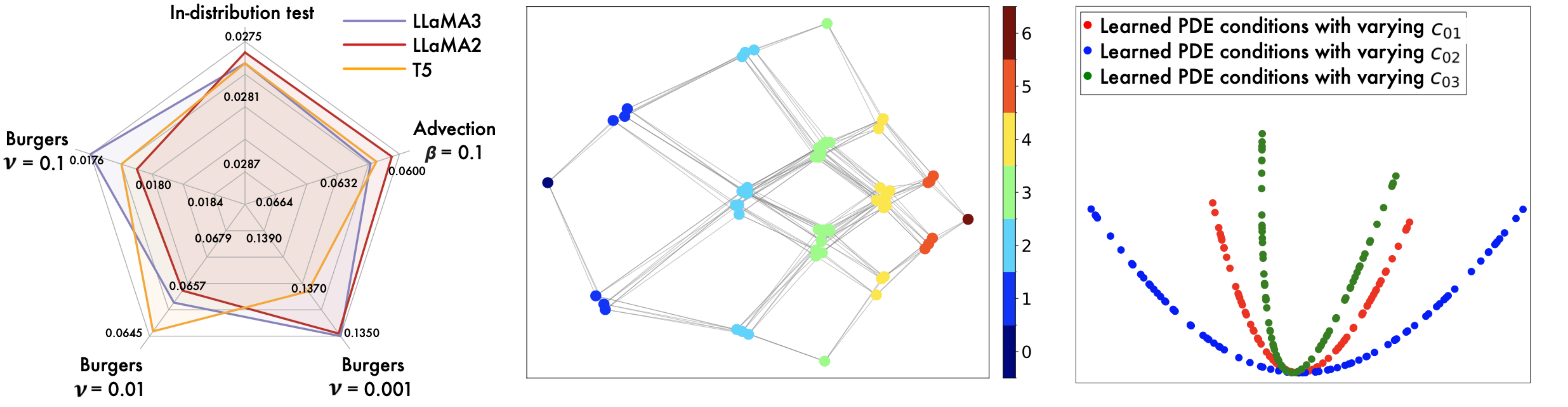


(a) Data Scalability (Samples) (b) Model Scalability (Parameters)

We progressively increase the **training data by 60 times** and **the model parameters by 21 times**, plotting the Relative L2 error on a log-log scale

Effect of LLM

Unisolver can utilize the prior knowledge of LLM to embed PDE symbolics, where **similar PDEs are embedded to closer representations**.



(a) Comparison of different LMs (b) PCA visualization of LLM embeddings (c) PCA visualization of learned PDE conditions

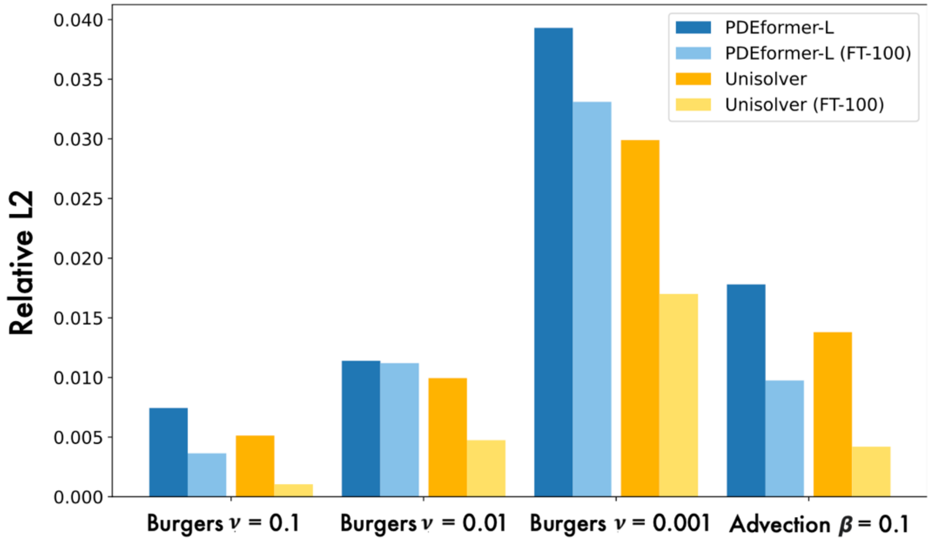
$$\partial_t u + f_0(u) + s(x) + \partial_x(f_1(u) - \kappa(x)\partial_x u) = 0,\ (x,t) \in [-1,1] \times [0,1].$$

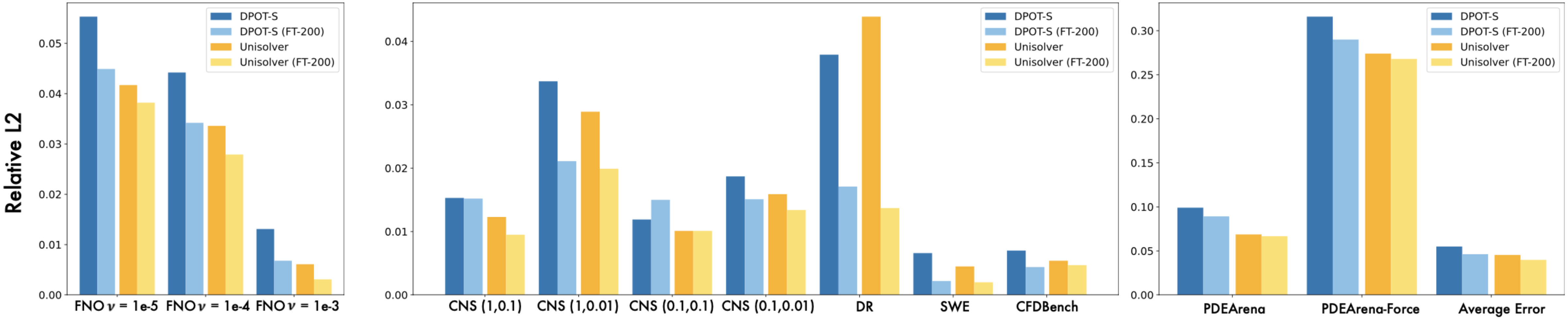
$$u(0,x) = g(x),\ x \in [-1,1].\quad f_i(u) = c_{i1}u + c_{i2}u^2 + c_{i3}u^3, i = 0,1.$$

Fine-tunning Performance

Finetune Unisolver trained from training sets and finetune the model with 20% training epochs.

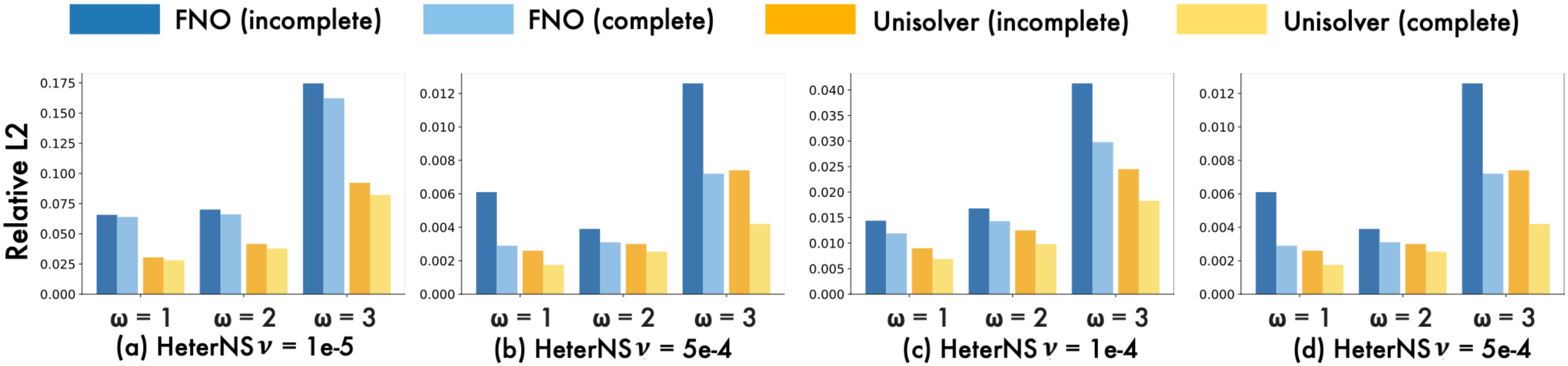
- ✓ Fast adaption to new PDEs
- ✓ Consistently improve model performance





(a) FNO-NS-ν (b) PDEBench & CFDBench (c) PDEArena & Average

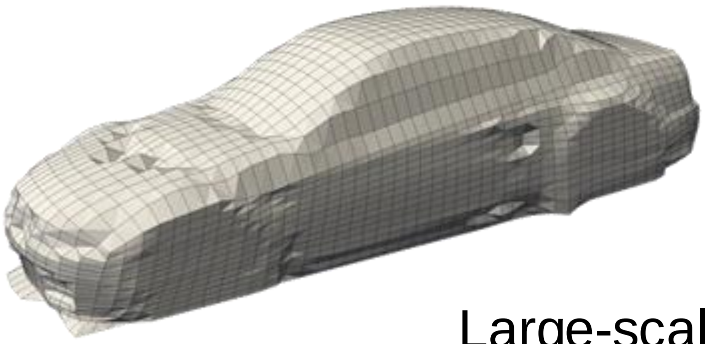
Experiments with Incomplete Components



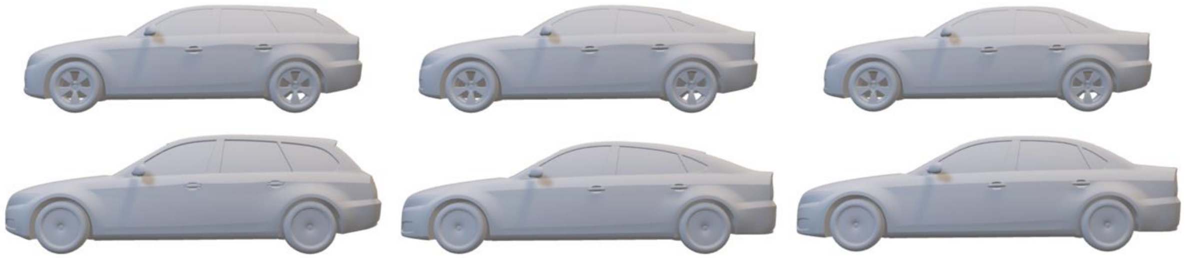
Under the incomplete PDE components (30% missing), Unisolver still the best.

But a complete set of components will further bring **21.6% average promotion**.

Our Exploration for Practical Neural PDE Solvers



Large-scale Meshes



Diverse PDEs, e.g. boundaries, coefficients, forces

1. Foundation Backbone:

   Transolver

2. Generalizable Models:

   Unisolver

https://github.com/thuml

https://ise.thss.tsinghua.edu.cn/~mlong/

Practical Neural PDE Solvers: Complex Geometries & OOD Generalization

Haixu Wu

School of Software, Tsinghua University

Oct 10, 2024

清華大學 Tsinghua University