From Transolver to Transolver-3:

Scaling Neural Solvers to Industrial-Scale Geometries

Haixu Wu

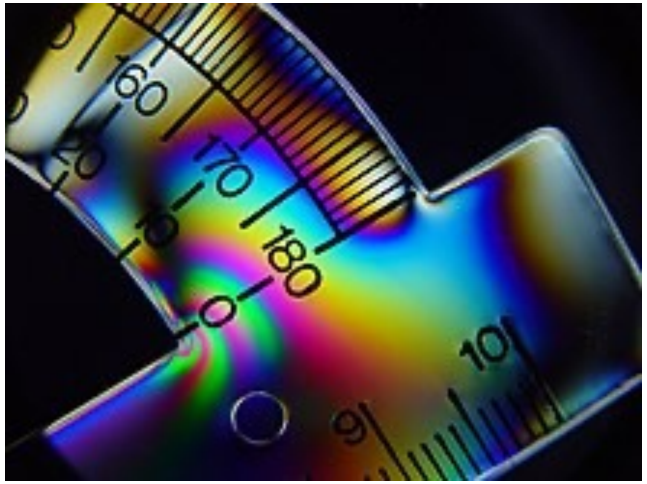
MIT CSAIL & THUML

Feb 04, 2026

Real-world Phenomena

Turbulence

Atmospheric circulation



Stress

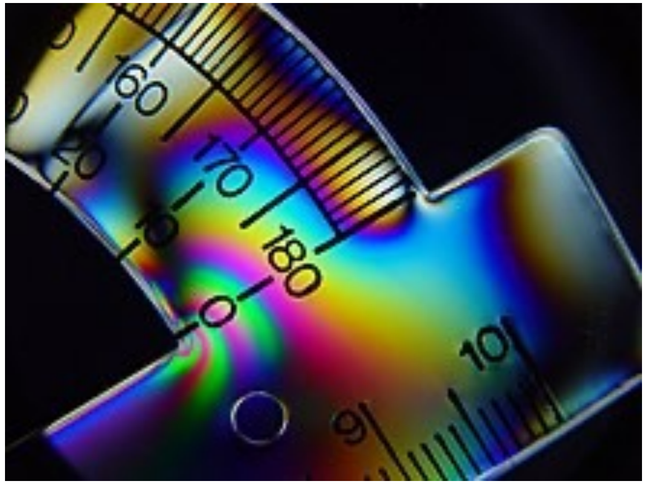
How to understand the world?

Images? Videos? World Model?

Real-world Phenomena

Turbulence

Atmospheric circulation

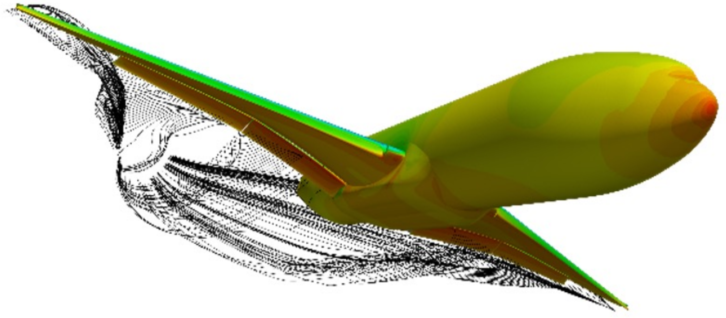


Stress

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

Partial Differential Equations

Extensive physics processes can be precisely described as PDEs.



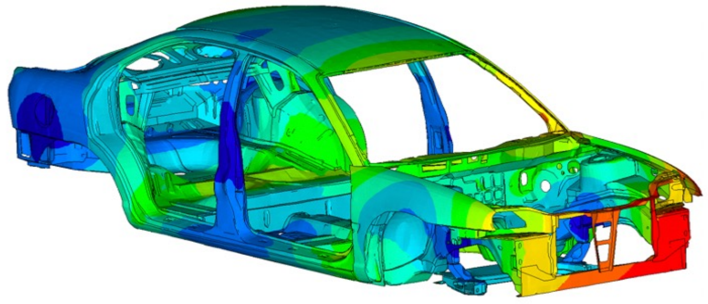
$$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}+\frac{\partial v_z}{\partial z}=0$$

$$\rho\left(\frac{\partial v_x}{\partial t}+v_x\frac{\partial v_x}{\partial x}+v_y\frac{\partial v_x}{\partial y}+v_z\frac{\partial v_x}{\partial z}\right)=-\frac{\partial P}{\partial x}+\mu\left(\frac{\partial^2 v_x}{\partial x^2}+\frac{\partial^2 v_x}{\partial y^2}+\frac{\partial^2 v_x}{\partial z^2}\right)+\rho g_x$$

$$\rho\left(\frac{\partial v_y}{\partial t}+v_x\frac{\partial v_y}{\partial x}+v_y\frac{\partial v_y}{\partial y}+v_z\frac{\partial v_y}{\partial z}\right)=-\frac{\partial P}{\partial y}+\mu\left(\frac{\partial^2 v_y}{\partial x^2}+\frac{\partial^2 v_y}{\partial y^2}+\frac{\partial^2 v_y}{\partial z^2}\right)+\rho g_y$$

$$\rho\left(\frac{\partial v_z}{\partial t}+v_x\frac{\partial v_z}{\partial x}+v_y\frac{\partial v_z}{\partial y}+v_z\frac{\partial v_z}{\partial z}\right)=-\frac{\partial P}{\partial z}+\mu\left(\frac{\partial^2 v_z}{\partial x^2}+\frac{\partial^2 v_z}{\partial y^2}+\frac{\partial^2 v_z}{\partial z^2}\right)+\rho g_z$$

3-D Navier-Stokes equations



$$\varepsilon_{xx}=\frac{\partial u_x}{\partial x},\quad \varepsilon_{yy}=\frac{\partial u_y}{\partial y},\quad \varepsilon_{zz}=\frac{\partial u_z}{\partial z}$$

$$\varepsilon_{xy}=\frac{1}{2}\left(\frac{\partial u_x}{\partial y}+\frac{\partial u_y}{\partial x}\right),\quad \varepsilon_{xz}=\frac{1}{2}\left(\frac{\partial u_x}{\partial z}+\frac{\partial u_z}{\partial x}\right),\quad \varepsilon_{yz}=\frac{1}{2}\left(\frac{\partial u_y}{\partial z}+\frac{\partial u_z}{\partial y}\right)$$

3-D Stress-Strain relations

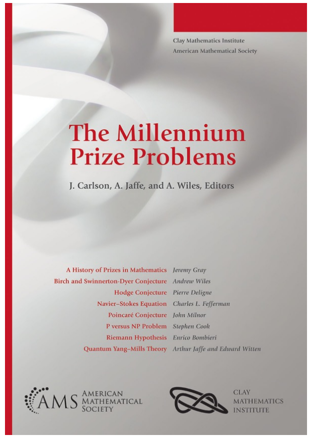
Difficulties in Solving PDEs

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

PDE Solvers

Classic Numerical Methods

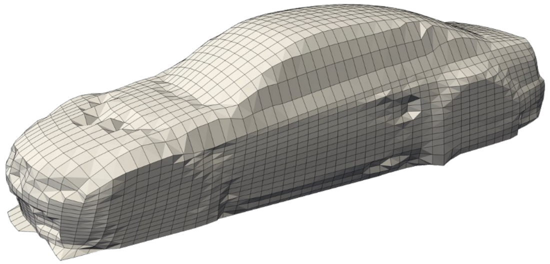
New Task → FEM, Spectral, etc → Results

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

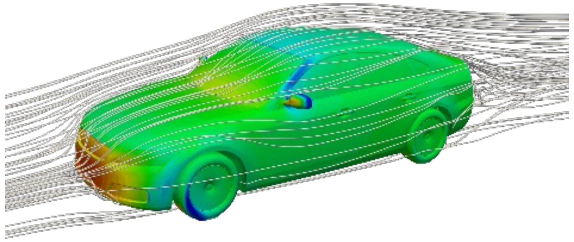
Stable vs. Slow and Discretized

Ansys · Dassault Systèmes · Siemens · Hexagon

Discretized Mesh

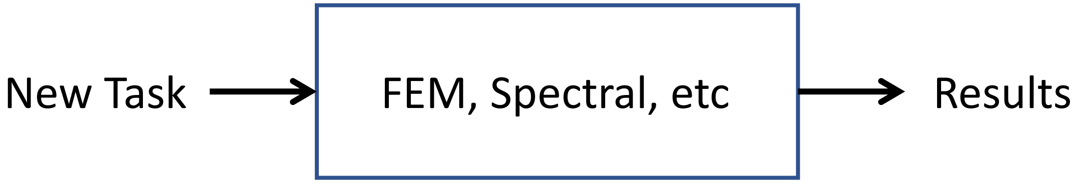


⇩ Days or even Months



PDE Solvers

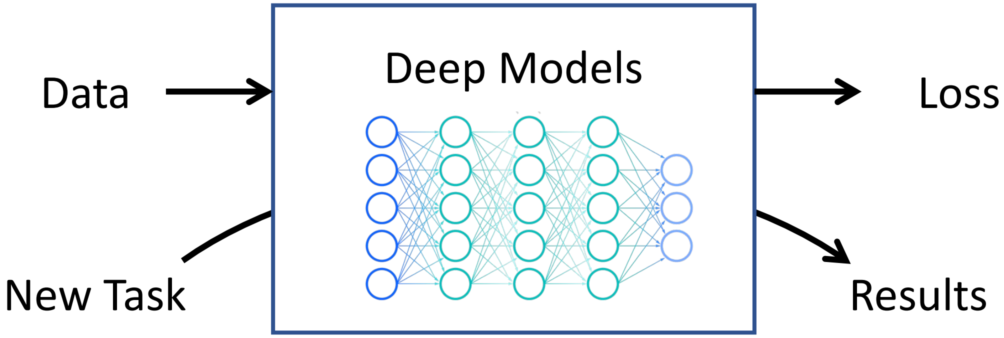
Classic Numerical Methods



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

Stable vs. Slow and Discretized

Neural PDE Solvers

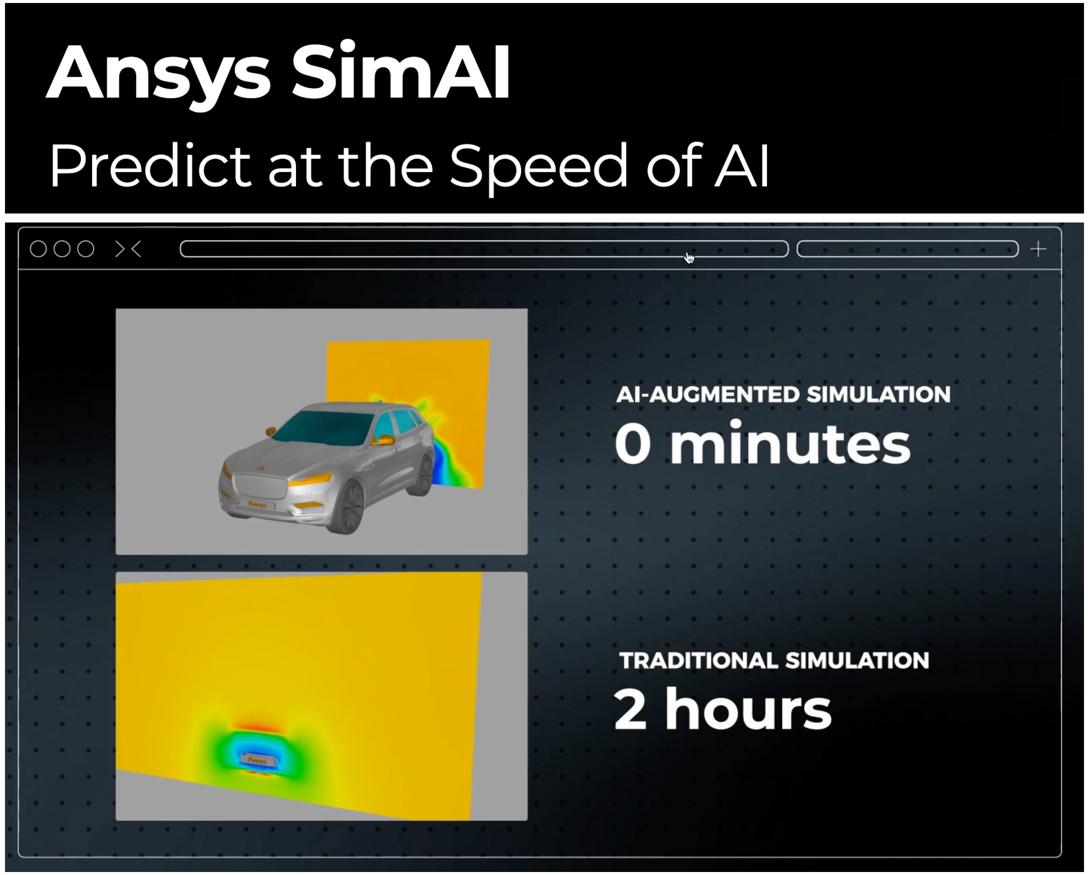


- Training once, inference a lot
- Each round needs several seconds

An efficient / precise surrogate tool

(Ideally)

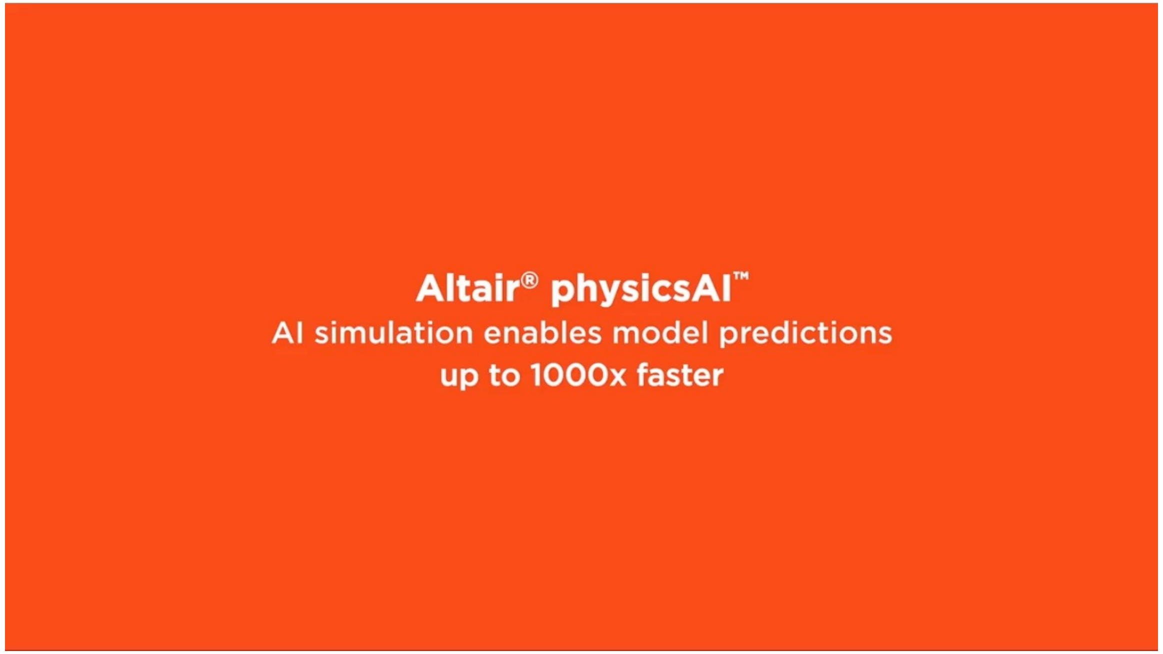
A Valuable Direction



https://www.ansys.com/products/simai

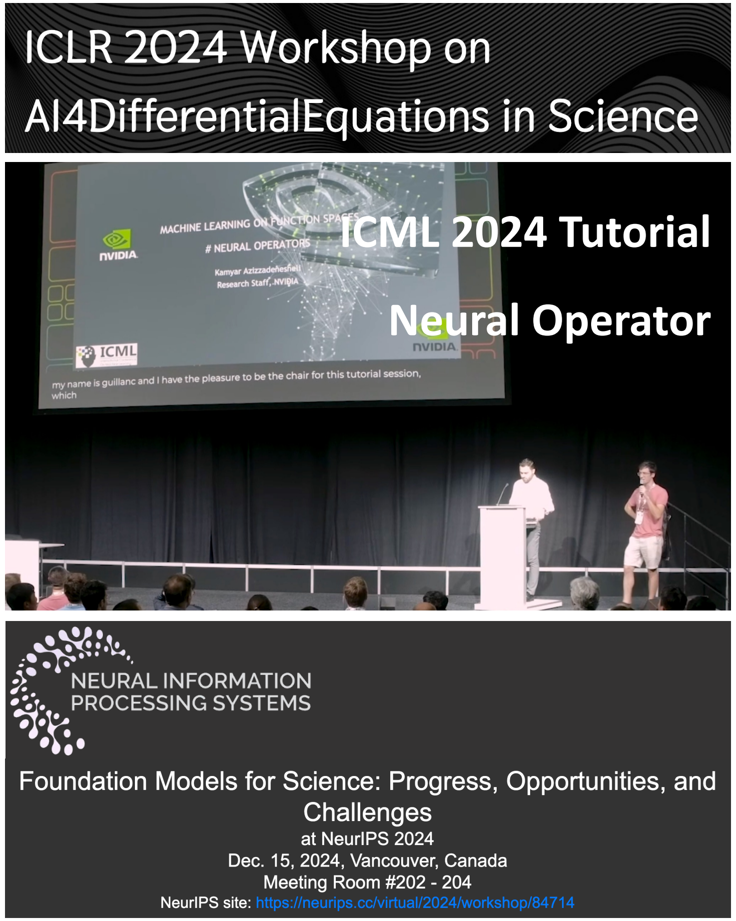
Altair® PhysicsAI™ Geometric Deep Learning

Better Design Insights Up to 1000x Faster than Solver Simulation



https://altair.com/physicsai

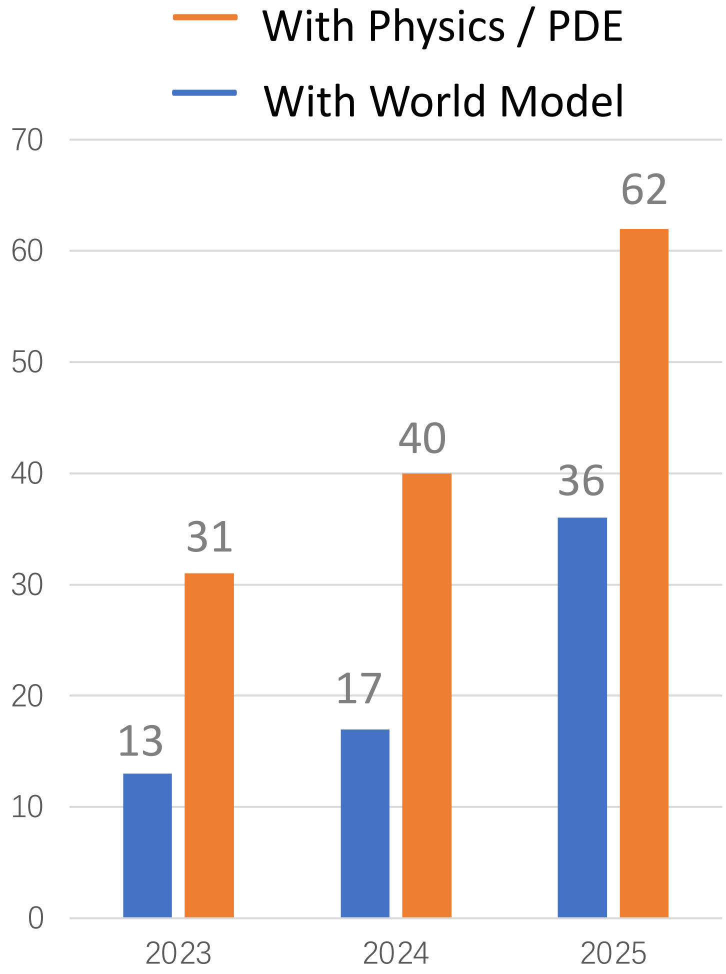
A Booming Direction



2024

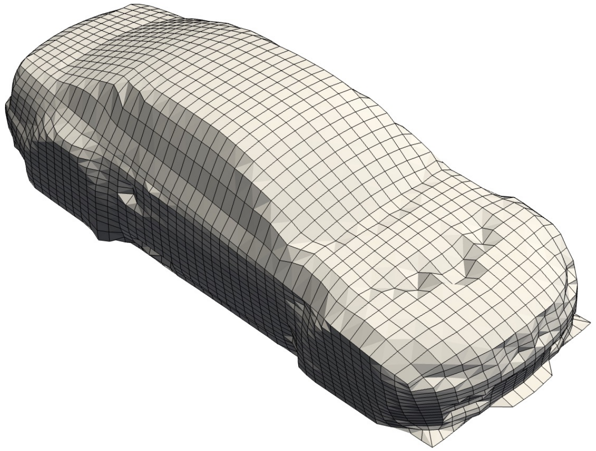
ICLR 2025

2025



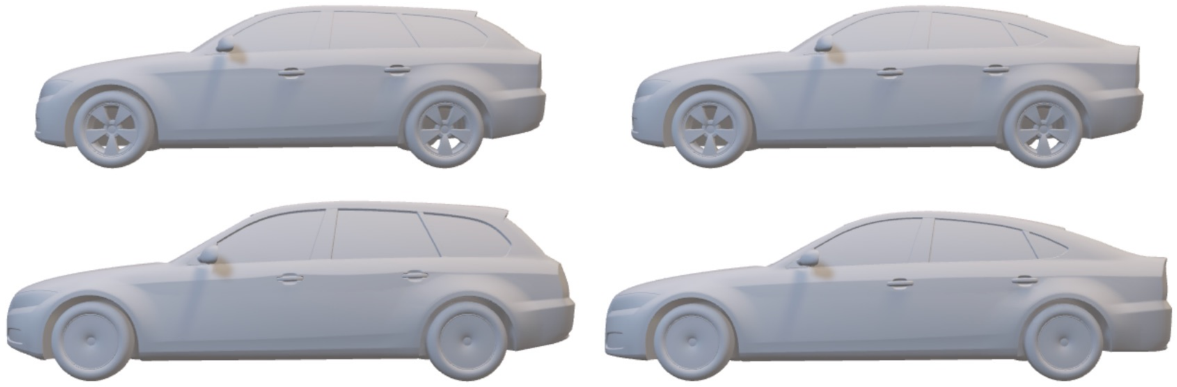
Accepted NeurIPS Papers

Towards Practical Neural PDE Solvers

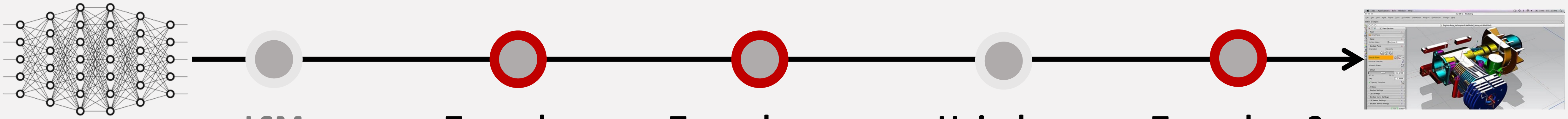


Complex Geometries

Large-scale Meshes



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



LSM (ICML 2023)

Transolver (ICML 2024)

Transolver++ (ICML 2025)

Unisolver (ICML 2025)

Transolver-3 (arXiv 2026)

ICML | 2024

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

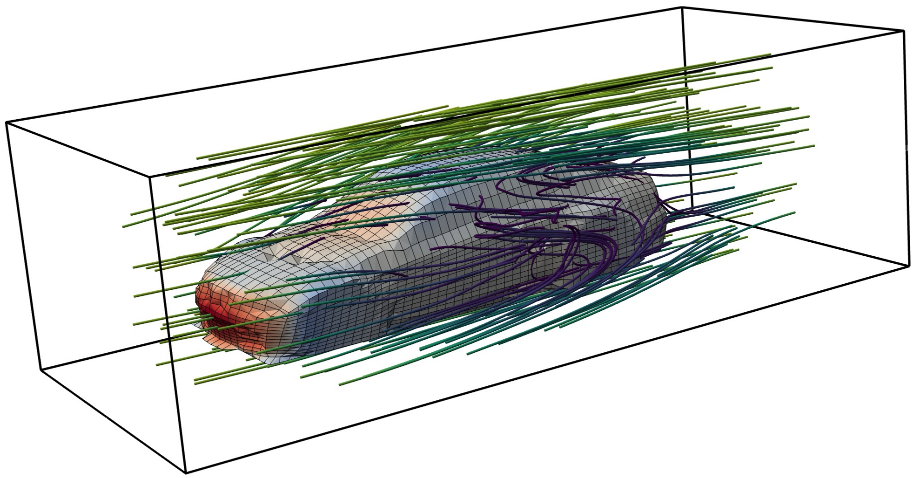
Haowen Wang

Jianmin Wang

Mingsheng Long

Code Link: https://github.com/thuml/Transolver

Challenges in Practical Industrial Design

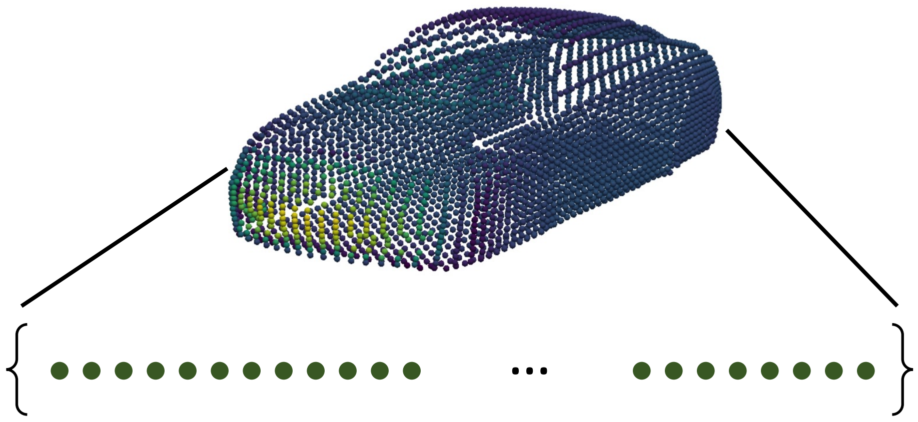


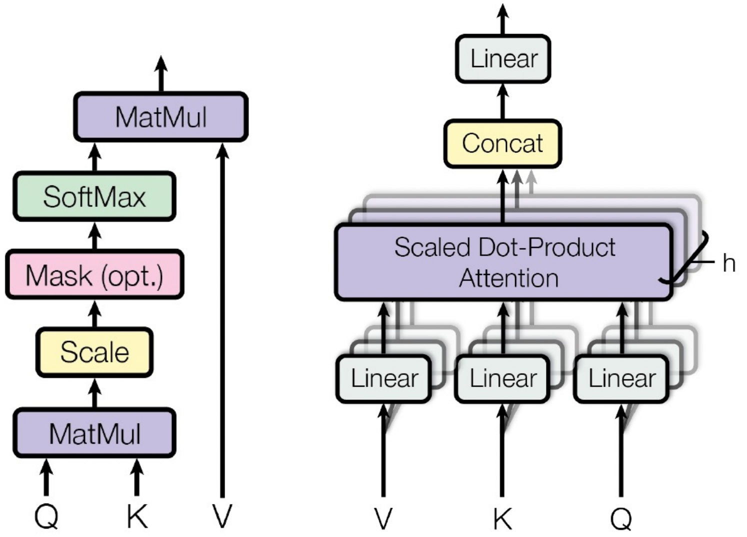
Task: Estimate the drag coefficient of a given shape:

Surrounding Wind & Surface Pressure

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

Transformer-based PDE Solvers





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

OFormer, Galerkin Transformer, GNOT, etc

Attention Mechanism as Global Integral

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:

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

Attention weight as kernel function

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

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

Dot-product Similarity

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:

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

Use the token sequence as an approximation of the integral

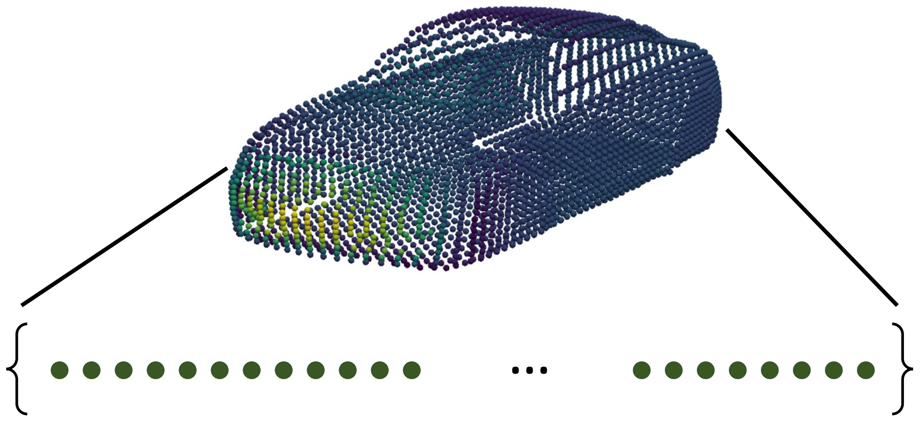
Applying the above equation to Eq. (7) and using the same approximation for the outer-integral, we have:

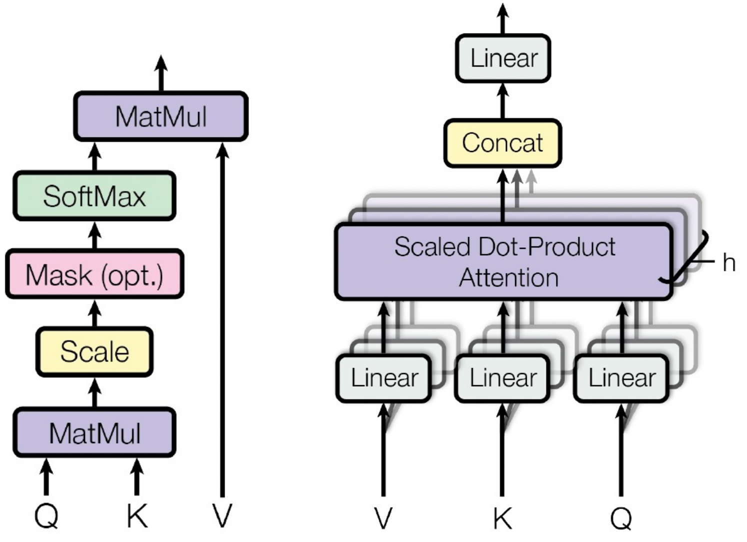
$$\mathcal{G}(\boldsymbol{u})(\mathbf{g}^*) \approx \sum_{i=1}^{N} \frac{\exp\left((\mathbf{W_q}\boldsymbol{u}(\mathbf{g}^*))\,(\mathbf{W_k}\boldsymbol{u}(\mathbf{g}_i))^\mathsf{T}\right)\mathbf{W_v}\boldsymbol{u}(\mathbf{g}_i)}{\sum_{j=1}^{N}\exp\left((\mathbf{W_q}\boldsymbol{u}(\mathbf{g}_j))\,(\mathbf{W_k}\boldsymbol{u}(\mathbf{g}_i))^\mathsf{T}\right)},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

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. □

Kovachki et al., Neural Operator: Learning Maps Between Function Spaces, JMLR 2022

Transformer-based PDE Solvers



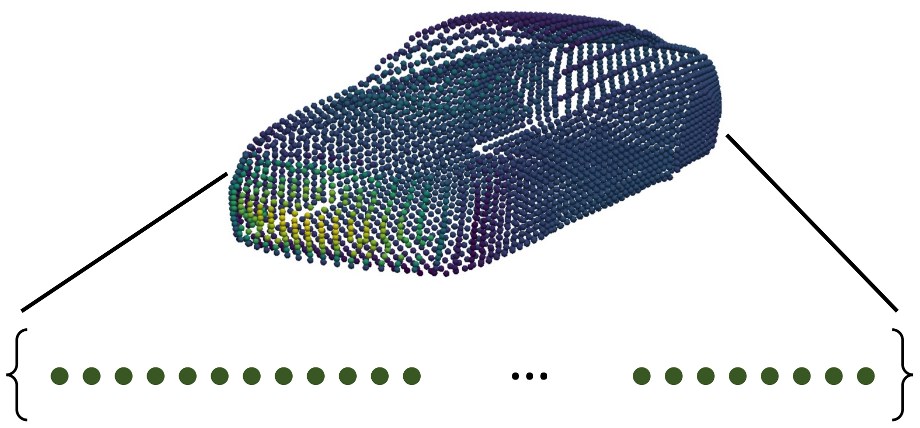


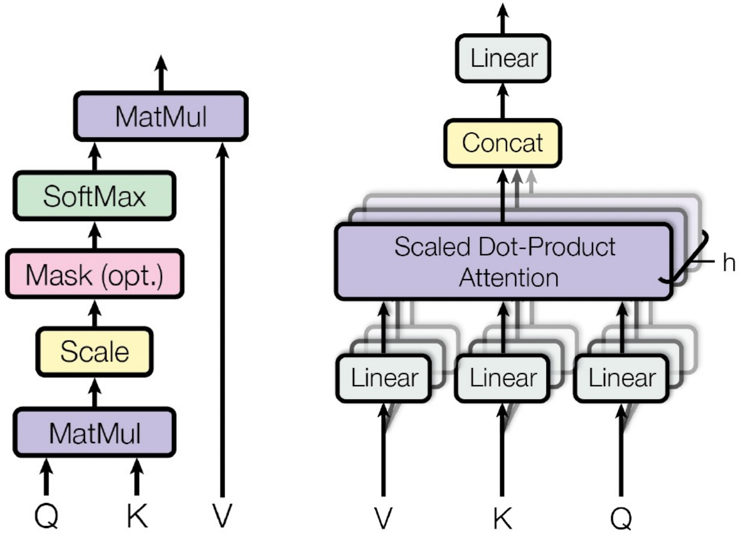
(1) Geometries as point sequences (2) Attention as Monte Carlo Integral

OFormer, Galerkin Transformer, GNOT, etc

1. **Quadratic complexity**
2. **Hard to capture physical correlations among massive points**

Transformer-based PDE Solvers





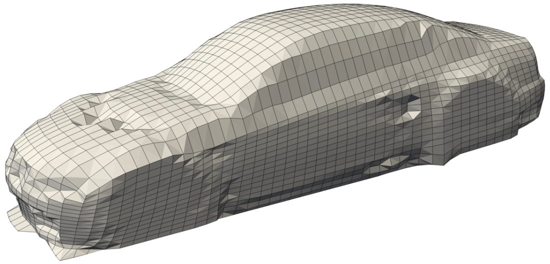
(1) Geometries as point sequences (2) Attention as Monte Carlo Integral

OFormer, Galerkin Transformer, GNOT, etc

How to efficiently capture physical correlations underlying discretized meshes

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

A Foundational Idea of Transolver



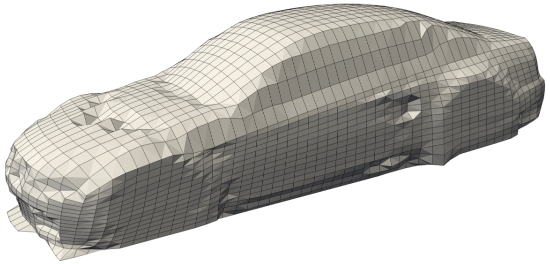
Discretized Domain

Previous Work

Being “trapped” to superficial and unwieldy meshes

Difficulties in Complexity, Geometry, Physics

A Foundational Idea of Transolver

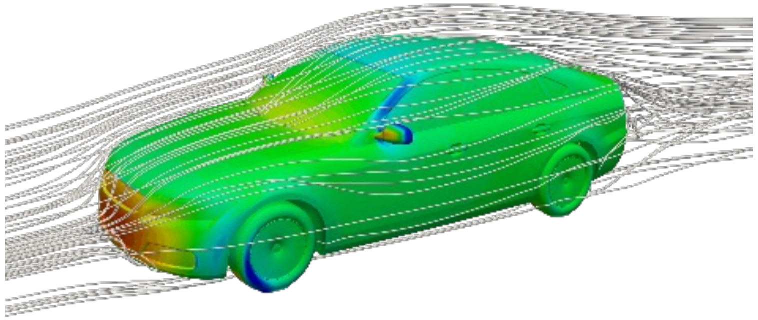


Discretized Domain

Previous Work

Being “trapped” to superficial and unwieldy meshes

Difficulties in Complexity, Geometry, Physics



Physics Domain

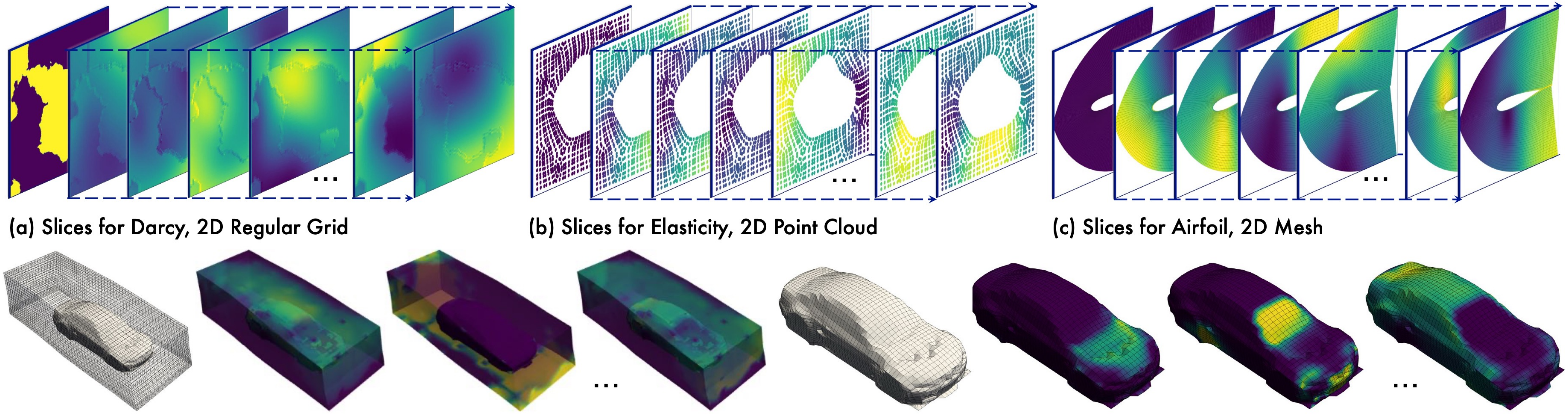
Transolver

Learning **intrinsic physical states** underlying

complex and large-scale geometries

Better Efficiency, Geometry, Physics Modeling

Learning Physical States

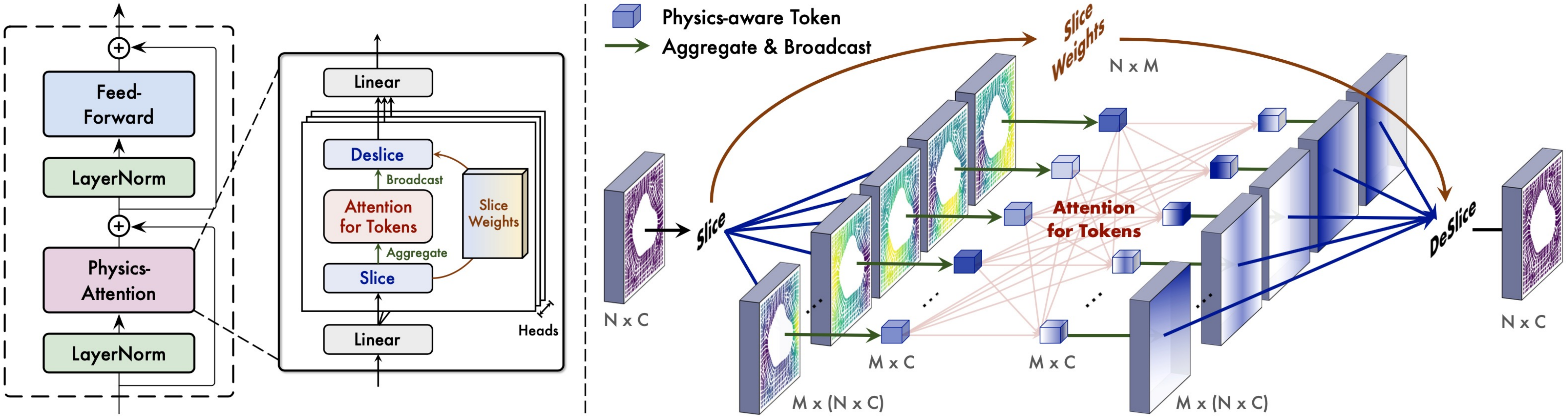


(a) Slices for Darcy, 2D Regular Grid (b) Slices for Elasticity, 2D Point Cloud (c) Slices for Airfoil, 2D Mesh

(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.

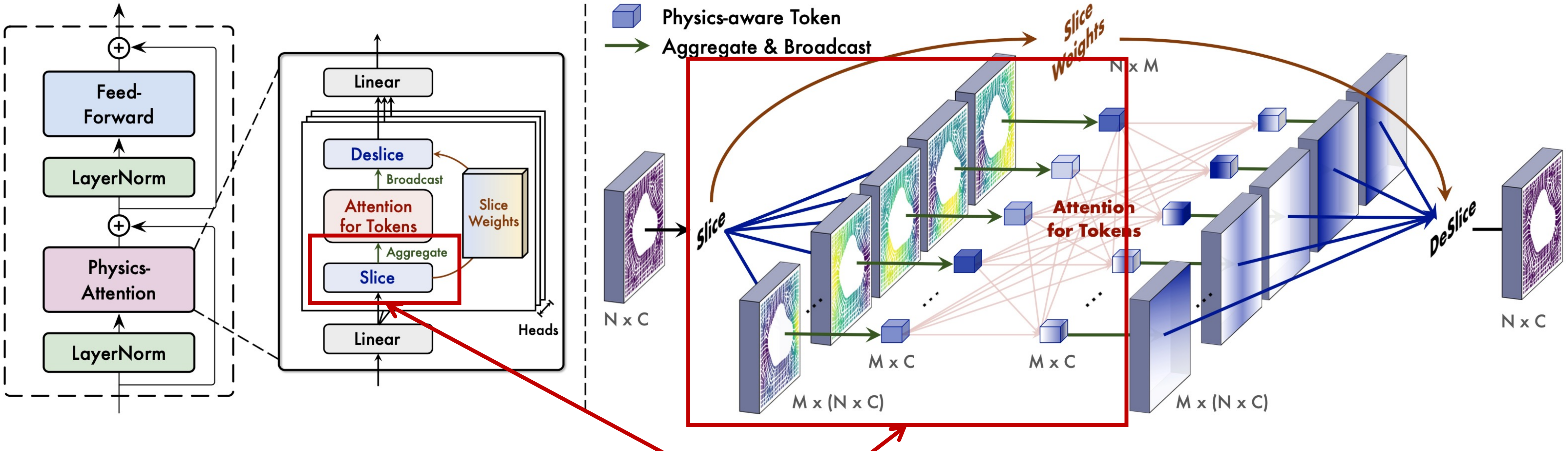
Overview of Transolver



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

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

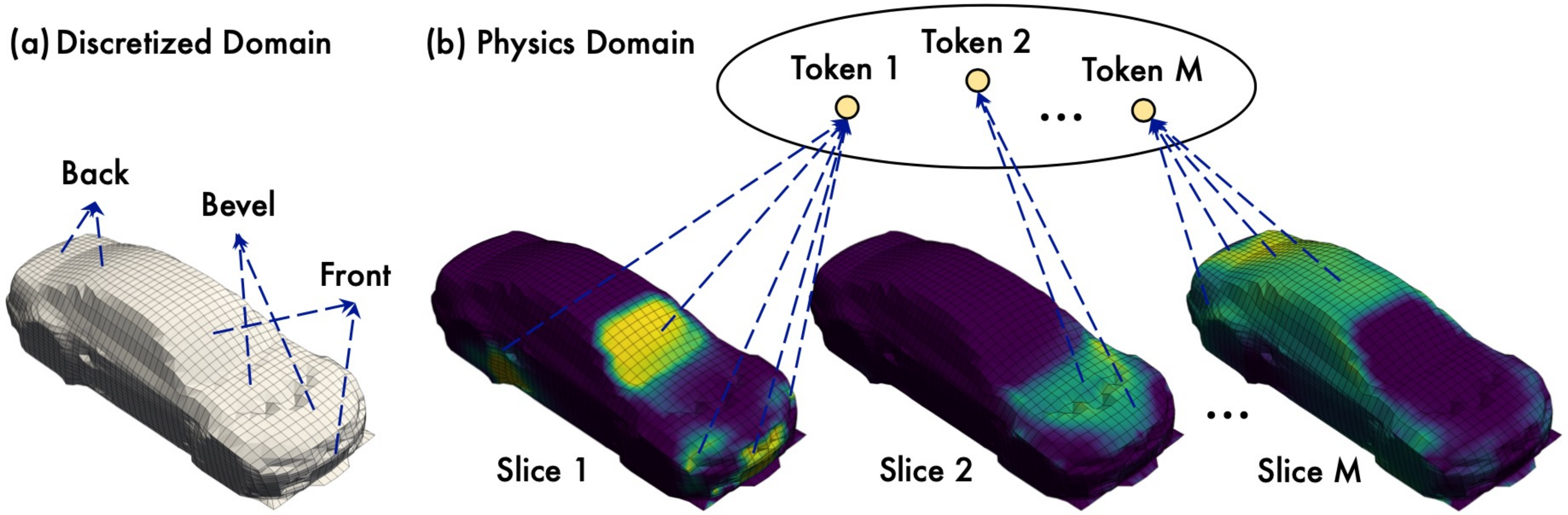
Step 1: Mesh → Physics



① Mesh → physics

To obtain physics-aware tokens

Learning Physics-aware Tokens



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

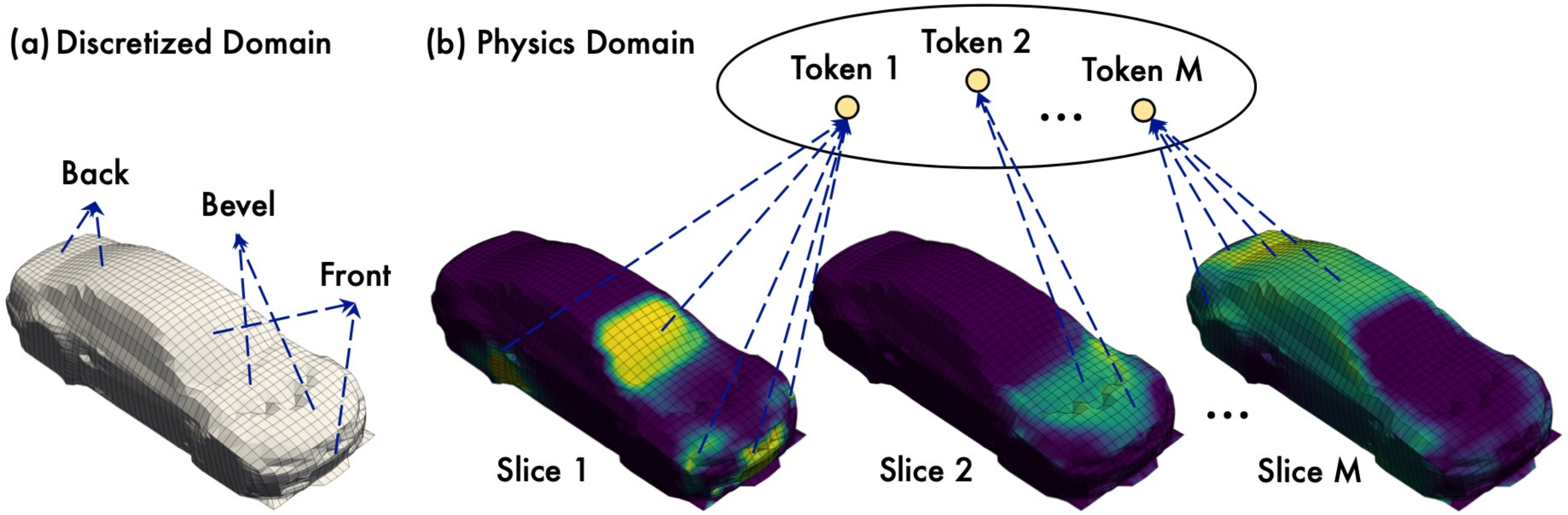
$$\{\mathbf{w}_i\}_{i=1}^{N} = \left\{\underline{\text{Softmax}}\left(\text{Project}\left(\mathbf{x}_i\right)\right)\right\}_{i=1}^{N}$$

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

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

Softmax for low-entropy slices

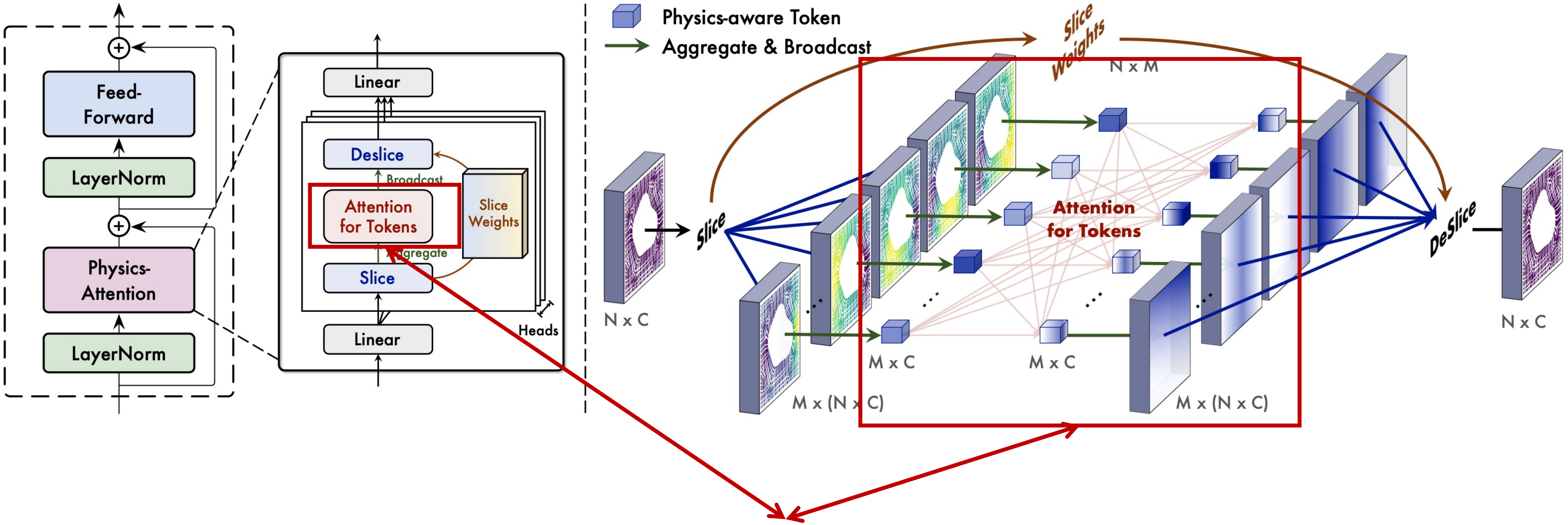
Learning Physics-aware Tokens



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

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

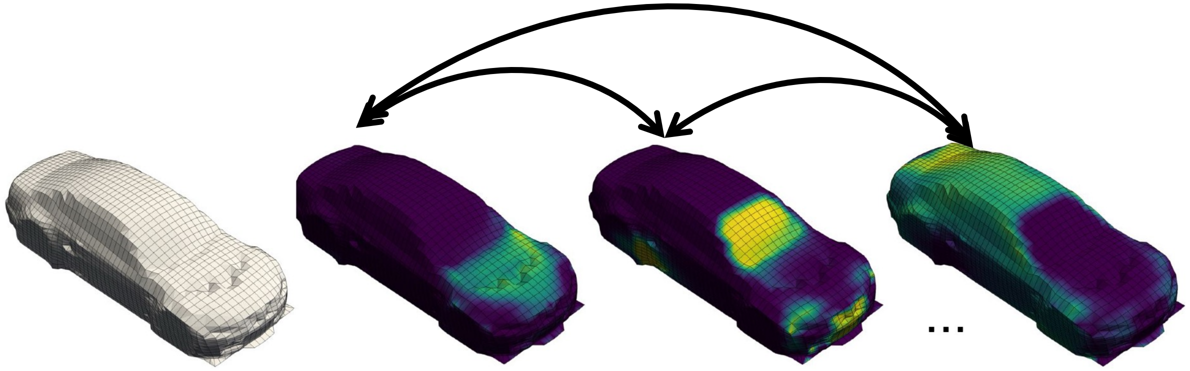
Step 2: Physics Interaction



② Attention among physics tokens

Approximate Integral to solve PDEs

Attention among physics tokens

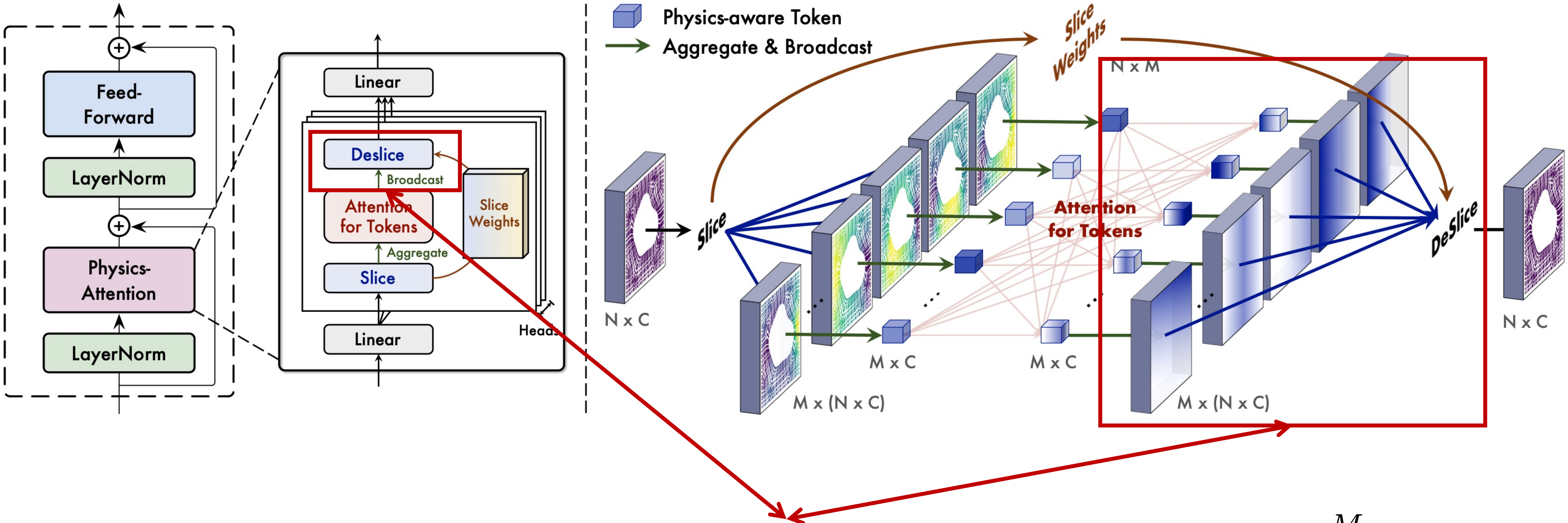


$$\mathbf{q}, \mathbf{k}, \mathbf{v} = \mathrm{Linear}(\underline{\mathbf{z}}), \quad \mathbf{z}' = \mathrm{Softmax}\left(\frac{\mathbf{q}\mathbf{k}^{\mathsf{T}}}{\sqrt{C}}\right)\mathbf{v}$$

Canonical attention among physics tokens

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

Step 3: Physics → Mesh



③ Physics → Mesh

Project physics information back to mesh

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

Slice weight

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 physics domain**

Is Physics-Attention still an input domain 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

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

Physics-Attention is still an <u>input domain</u> integral.

$$= \int_{\Omega_{\mathrm{s}}} \kappa_{\mathrm{ms}}\left(\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}\right)\boldsymbol{u}_{\mathrm{s}}\left(\boldsymbol{\xi}_{\mathrm{s}}\right)\mathrm{d}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}) \qquad (\kappa_{\mathrm{ms}}(\cdot,\cdot):\Omega\times\Omega_{\mathrm{s}}\to\mathbb{R}^{C\times C}\text{ is a kernel function})$$

$$= \int_{\Omega_{\mathrm{s}}} \kappa_{\mathrm{ms}}\left(\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}\right)\boldsymbol{u}_{\mathrm{s}}\left(\boldsymbol{\xi}_{\mathrm{s}}\right)|\det(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}))|\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}$$

$$= \int_{\Omega_{\mathrm{s}}} \left(\frac{\int_{\Omega_{\mathrm{s}}} w_{\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}'}\kappa_{\mathrm{ss}}\left(\boldsymbol{\xi}_{\mathrm{s}}',\boldsymbol{\xi}_{\mathrm{s}}\right)\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}'}{\int_{\Omega_{\mathrm{s}}} w_{\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}'}\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}'}\right)\boldsymbol{u}_{\mathrm{s}}\left(\boldsymbol{\xi}_{\mathrm{s}}\right)|\det(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}))|\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}} \qquad (\kappa_{\mathrm{ms}}\text{ is a linear combination of }\kappa_{\mathrm{ss}}\text{ with weights }w_{*,*})$$

$$= \int_{\Omega_{\mathrm{s}}} \underbrace{w_{\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}'}}_{\text{DeSlice}} \int_{\Omega_{\mathrm{s}}} \underbrace{\kappa_{\mathrm{ss}}\left(\boldsymbol{\xi}_{\mathrm{s}}',\boldsymbol{\xi}_{\mathrm{s}}\right)}_{\text{Attention among slice tokens}} \underbrace{\boldsymbol{u}_{\mathrm{s}}\left(\boldsymbol{\xi}_{\mathrm{s}}\right)}_{\text{Slice token}}|\det(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}))|\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}' \qquad \left(\text{Suppose that }\int_{\Omega_{\mathrm{s}}} w_{\mathbf{g},\boldsymbol{\xi}_{\mathrm{s}}'}\mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}'=1\right)$$

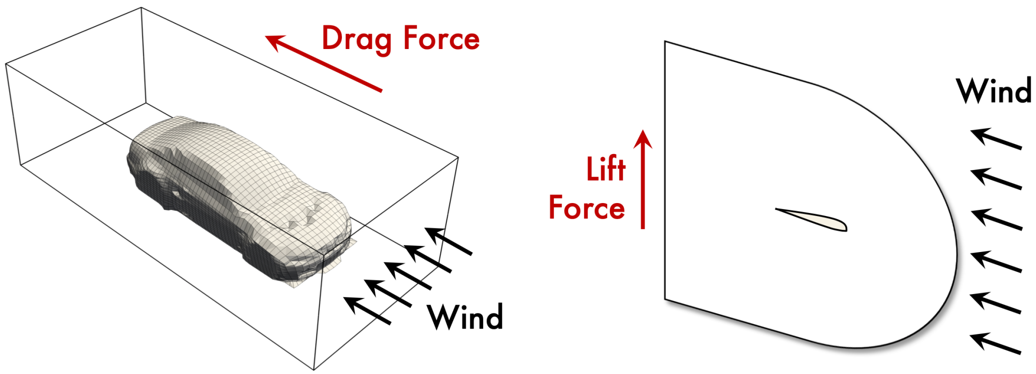
All the designs can be directly derived.

$$\approx \underbrace{\sum_{j=1}^{M}\mathbf{w}_{i,j}}_{\text{Eq. (4)}}\underbrace{\sum_{t=1}^{M}\frac{\exp\left(\left(\mathbf{W}_{\mathbf{q}}\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},j})\right)\left(\mathbf{W}_{\mathbf{k}}\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},t})\right)^{\mathsf{T}}/\tau\right)}{\sum_{p=1}^{M}\exp\left(\left(\mathbf{W}_{\mathbf{q}}\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},j})\right)\left(\mathbf{W}_{\mathbf{k}}\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},p})\right)^{\mathsf{T}}/\tau\right)}\mathbf{W}_{\mathbf{v}}}_{\text{Eq. (3)}}\underbrace{\left(\frac{\sum_{p=1}^{N}\mathbf{w}_{p,t}\boldsymbol{u}(\mathbf{g}_p)}{\sum_{p=1}^{N}\mathbf{w}_{p,t}}\right)}_{\text{Eq. (2)}} \qquad (\text{Lemma A.1})$$

$$= \sum_{j=1}^{M}\mathbf{w}_{i,j}\sum_{t=1}^{M}\frac{\exp(\mathbf{q}_j\mathbf{k}_t^{\mathsf{T}}/\tau)}{\sum_{p=1}^{M}\exp(\mathbf{q}_j\mathbf{k}_p^{\mathsf{T}}/\tau)}\mathbf{v}_t,$$

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	0.0059	0.0050	0.1535	0.0065
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	0.0017	0.0183	0.0168	0.1705	0.0124
GNOT (Hao et al., 2023)	0.0086	0.0336	0.0076	0.0047	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	0.1195	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%

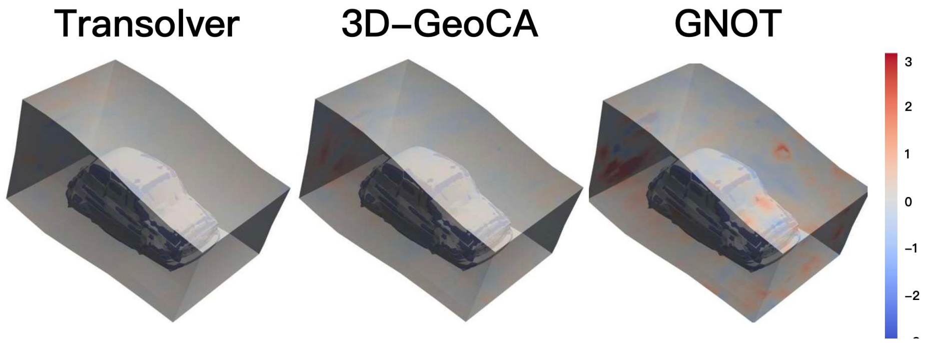
Transolver achieves 22% error reduction over the second-best model

Car and Airfoil Design

Model capability in "ranking" designs

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

模型 *	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 [197]	0.0461	0.1050	0.0270	0.9695	0.0087	0.0184	0.1476	0.9964
PointNet [196]	0.0494	0.1104	0.0298	0.9583	0.0253	0.0996	0.1973	0.9919
Graph U-Net [206]	0.0471	0.1102	0.0226	0.9725	0.0076	0.0144	0.1677	0.9949
MeshGraphNet [198]	0.0354	0.0781	0.0168	0.9840	0.0214	0.0387	0.2252	0.9945
GNO [80]	0.0383	0.0815	0.0172	0.9834	0.0269	0.0405	0.2016	0.9938
Galerkin [203]	0.0339	0.0878	0.0179	0.9764	0.0074	0.0159	0.2336	0.9951
geo-FNO [192]	0.1670	0.2378	0.0664	0.8280	0.0361	0.0301	0.6161	0.9257
GNOT [85]	0.0329	0.0798	0.0178	0.9833	0.0049	0.0152	0.1992	0.9942
GINO [199]	0.0386	0.0810	0.0184	0.9826	0.0297	0.0482	0.1821	0.9958
3D-GeoCA [193]	0.0319	0.0779	0.0159	0.9842	/	/	/	/
Transolver	**0.0207**	**0.0745**	**0.0103**	**0.9935**	**0.0037**	**0.0142**	**0.1030**	**0.9978**

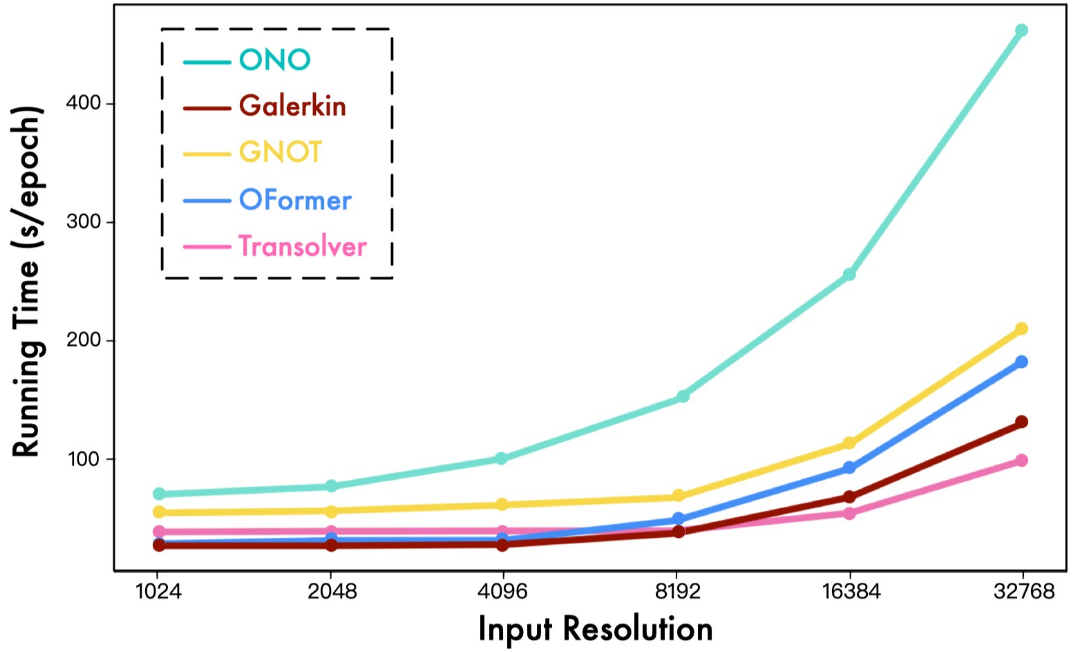


Surrounding Velocity Error Map

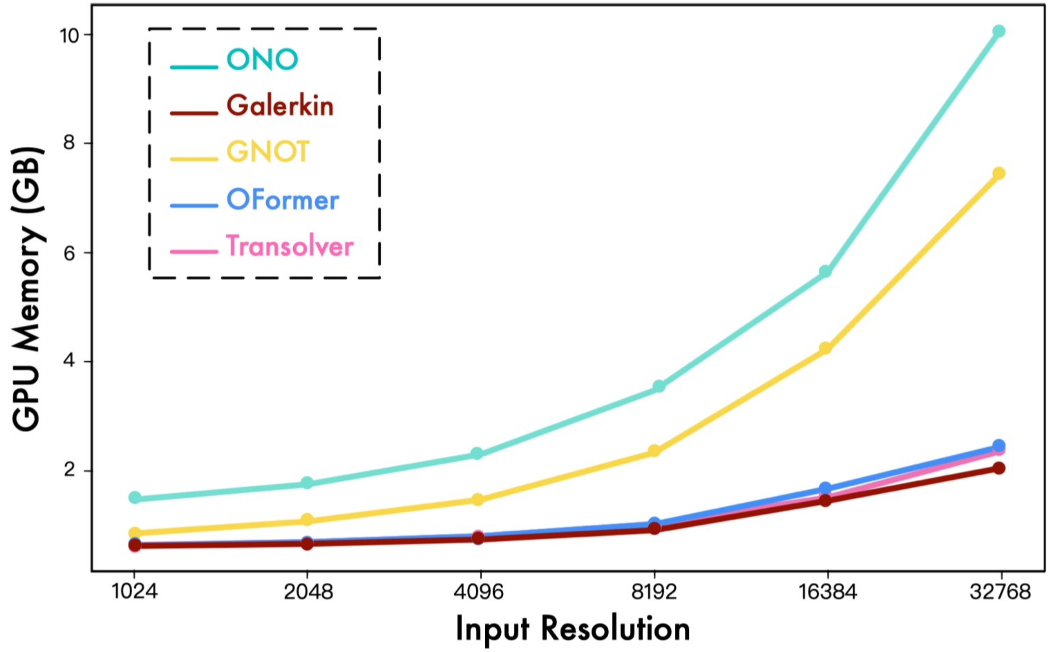
Surface Pressure Error Map

Efficiency

Running Time



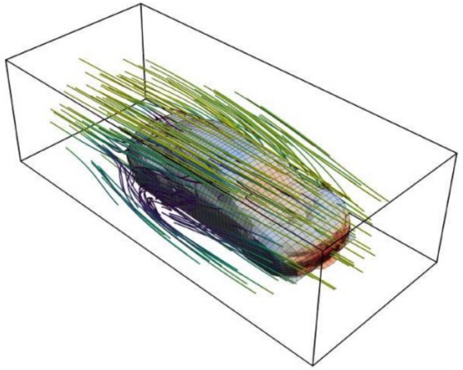
GPU Memory

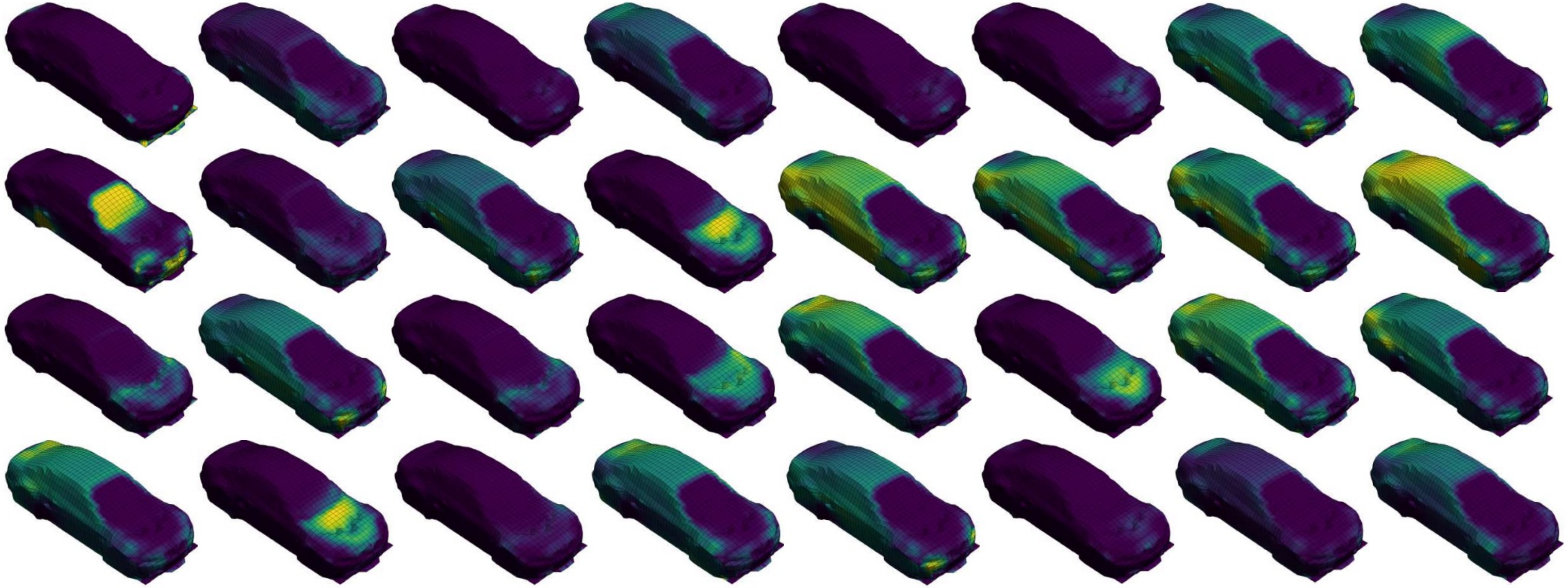


Favorable efficiency and performance balance

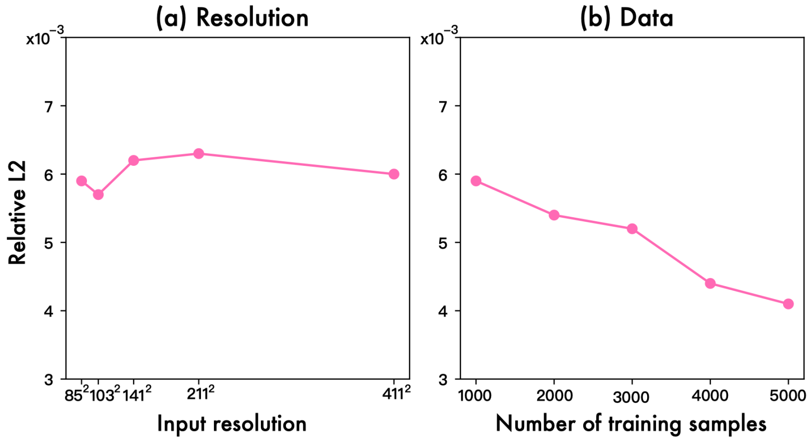
Transolver is faster than linear Transformers in large-scale meshes.

Physical States Visualization

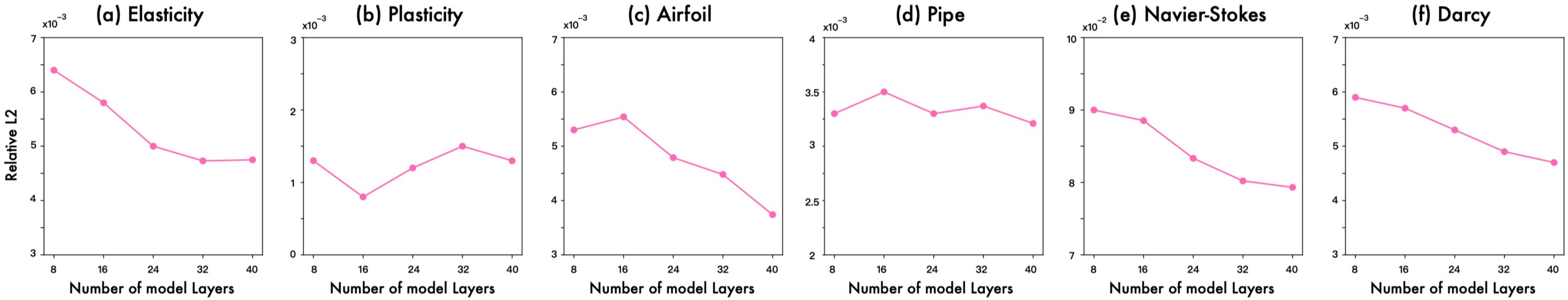




Pursuing PDE Foundation Models: Scalability

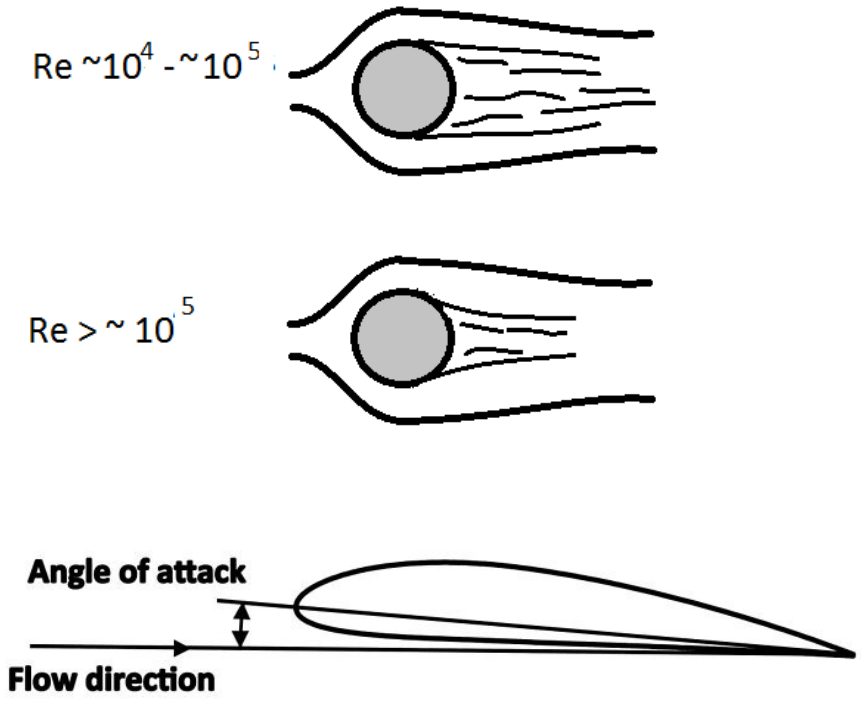


1. **Resolution:** Consistent performance at varied scales
2. **Data:** Benefiting from larger training data
3. **Parameter:** Benefiting from more parameters



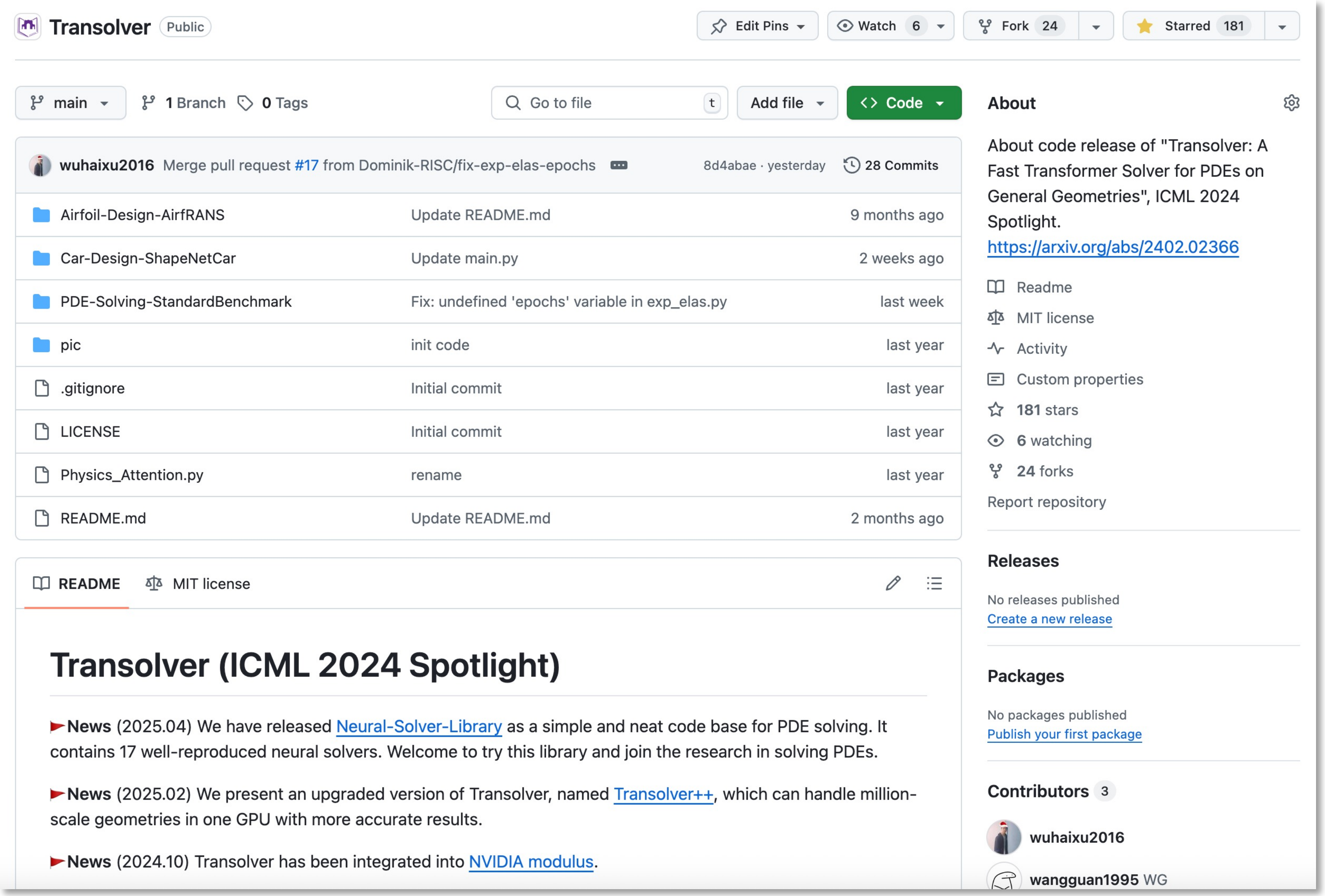
Pursuing PDE Foundation Models: Generalization

Models	OOD Reynolds $C_L \downarrow$	OOD Reynolds $\rho_L \uparrow$	OOD Angles $C_L \downarrow$	OOD Angles $\rho_L \uparrow$
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**



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

Open-Source Code



NVIDIA

physicsnemo

Code for Transolver in Physicsnemo

Code for Transolver

Code Link: *https://github.com/thuml/Transolver*

NVIDIA PhysicsNeMo

NVIDIA PhysicsNeMo Framework 25.08

Logging and Checkpointing
Model Architectures
PhysicsNeMo Distributed
Physics-guided
Performance
Data Curation
Model Evaluation and Inference
Symbolic Abstractions

Examples

PhysicsNeMo Examples Catalog

Library Documentation

PhysicsNeMo
PhysicsNeMo Sym
PhysicsNeMo Curator
PhysicsNeMo CFD
Earth2Studio

Resources

Customizing PhysicsNeMo

Releases

New features/Highlights v25.08

Features and Enhancements

- GNNs: Support for Pytorch Geometric and MeshGraphNet performance optimizations, between 1.5x to 2x speedup with float16, bfloat16 for meshes > 200k nodes.
- Transformers: Transolver performance optimization
- DoMINO fine-tuning.
- Updated DoMINO training recipe:
  - Physics informed DoMINO
  - Configure as many global parameters as needed
- Error quantification for external aerodynamics
- Data curation enhancements
- Mixture of experts for external aerodynamics.

Recipes and Examples

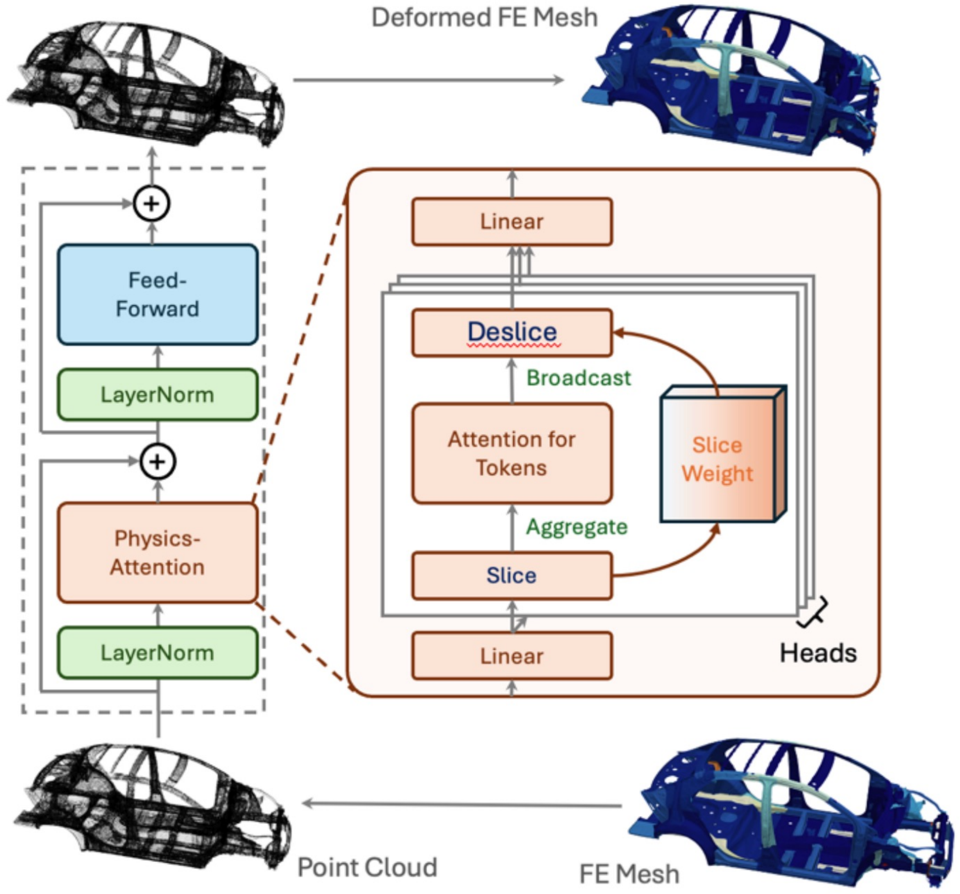
- Reference workflow for design sensitivity analysis using AI surrogates.
- Denoising Pre-trained Operator Transformer samples.
- FWI sample

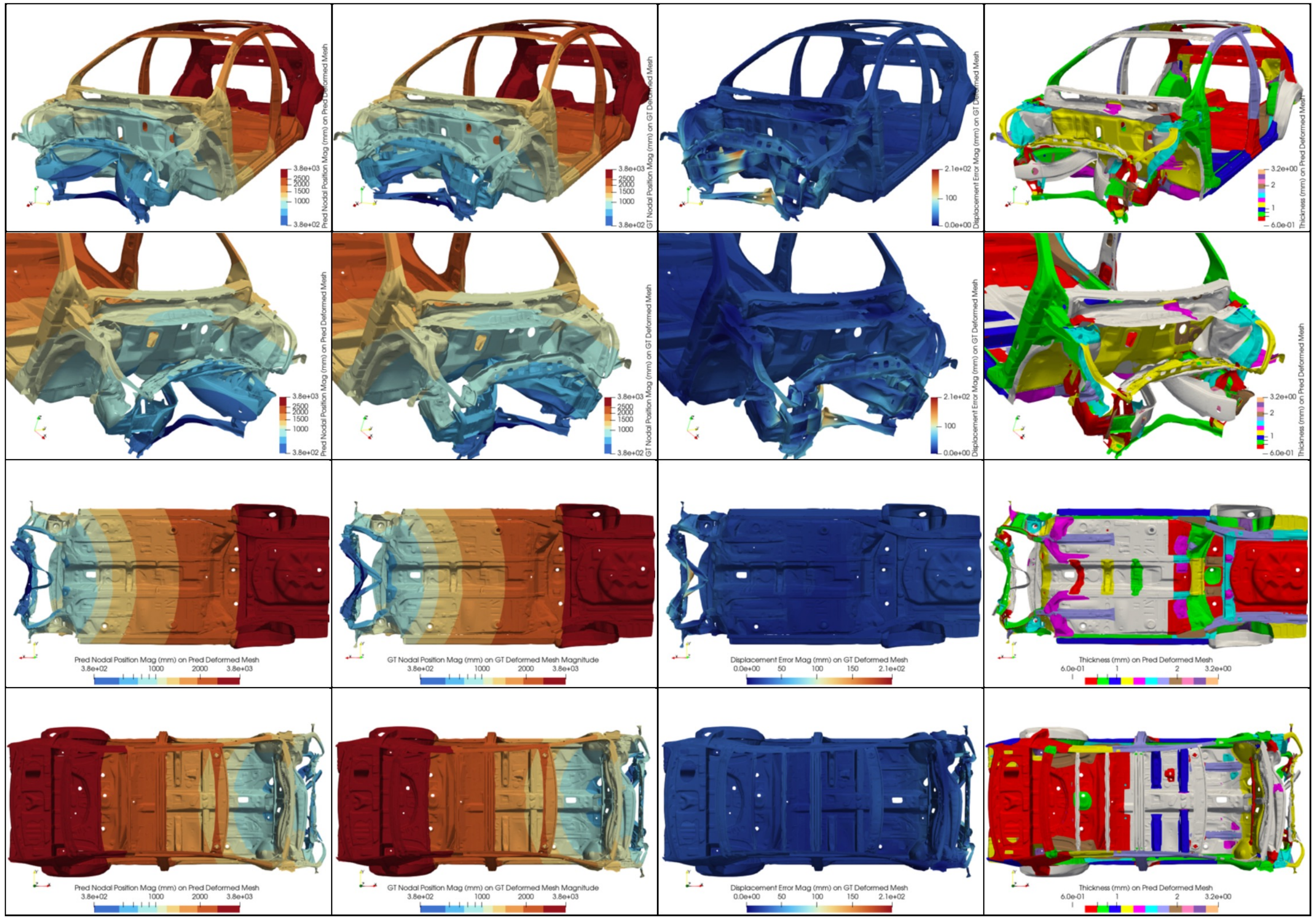
physicsnemo

"The Transolver model is a **promising**, transformer-based model that **produces high-quality predictions** for CFD surrogate simulations."

https://docs.nvidia.com/physicsnemo/25.08/physicsnemo/examples/cfd/external_aerodynamics/transolver/README.html

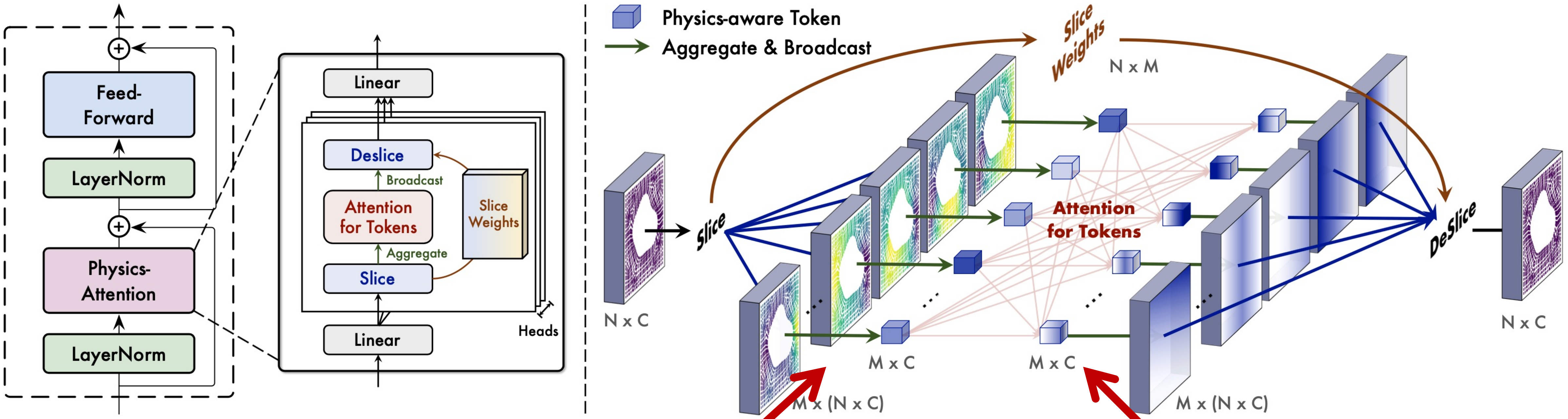
NVIDIA PhysicsNeMo





Nabian et al., Automotive Crash Dynamics Modeling Accelerated with Machine Learning, arXiv 2025

“Magic Design” in Transolver



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

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

Why adopt the global weighted sum?

Support Transolver++

Why reuse slice weights?

Support Transolver-3

ICML | 2025

The Forty-second International Conference on Machine Learning

Transolver++: An Accurate Neural Solver for PDEs on Million-Scale Geometries

Huakun Luo [* 1] **Haixu Wu** [* 1] **Hang Zhou** [1] **Lanxiang Xing** [1] **Yichen Di** [1] **Jianmin Wang** [1] **Mingsheng Long** [1]

Huakun Luo

Haixu Wu

Hang Zhou

Lanxiang Xing

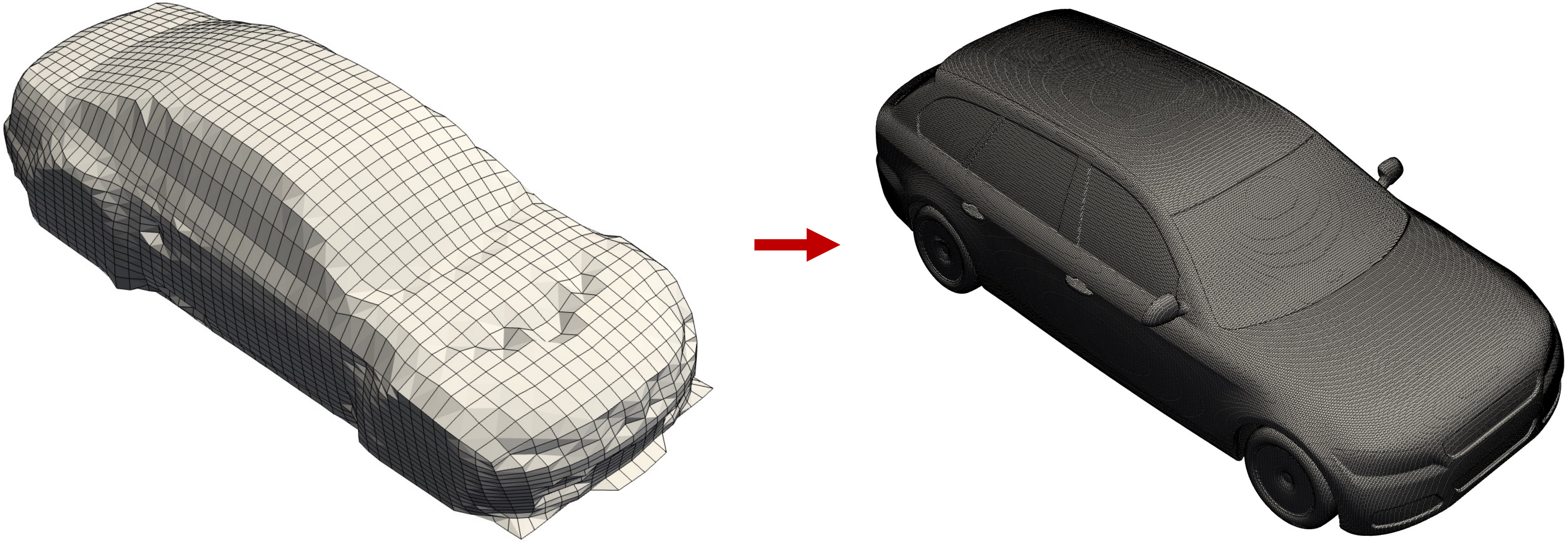
Yichen Di

Jianmin Wang

Mingsheng Long

Code Link: *https://github.com/thuml/Transolver_plus*

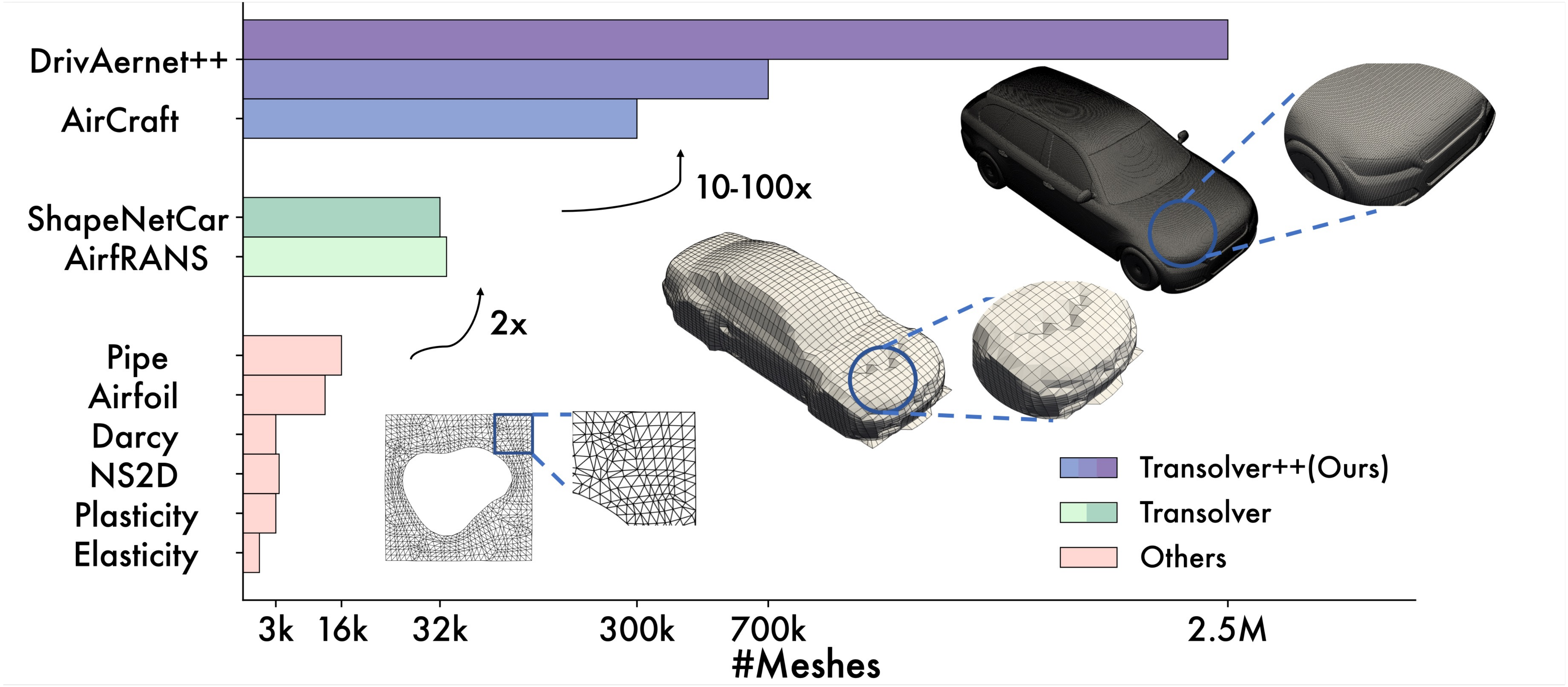
Extremely Large Geometries



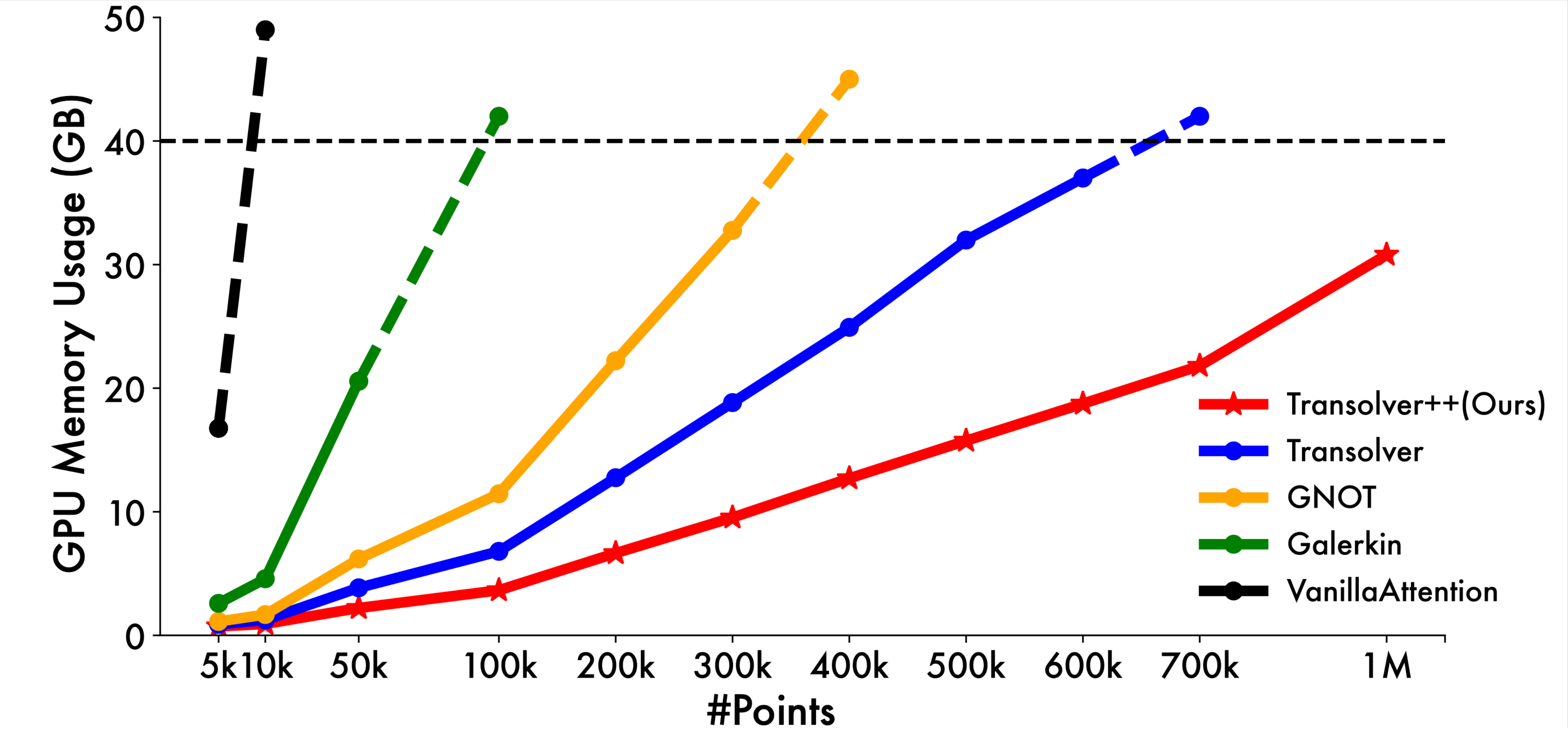
32k Mesh Points

2.5M Mesh Points

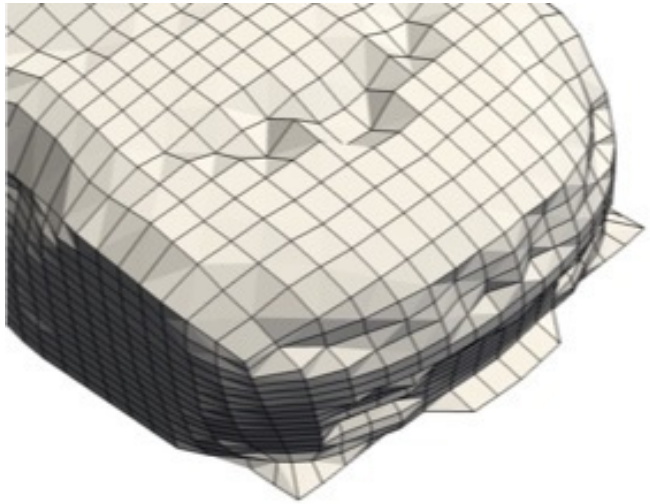
10-100x Larger than Previous Benchmarks

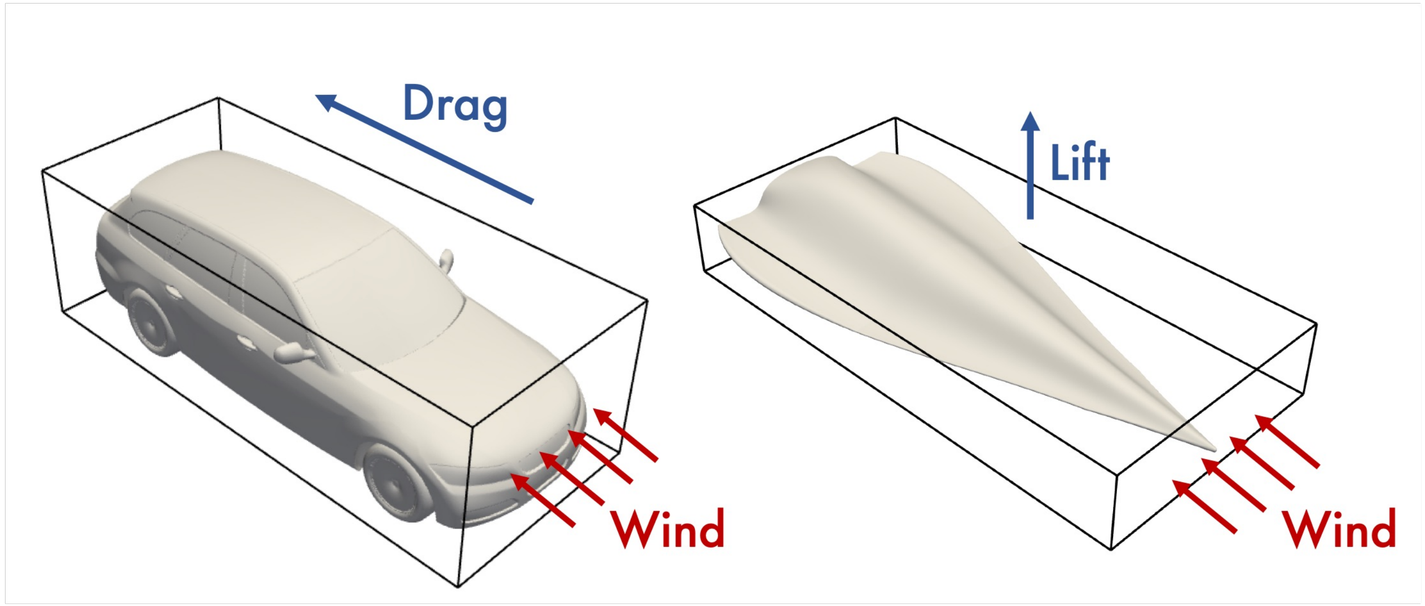


Transolver++: Enable PDE Solving in Million-Scale Geometries



Difficulties on Applicability





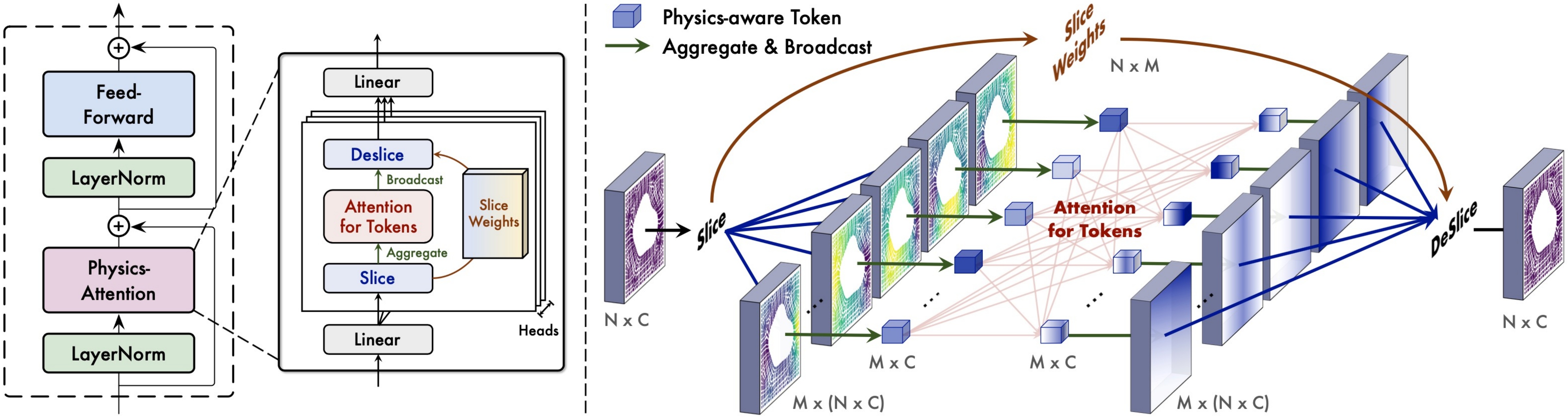
Large Geometrics In real-world applications

1. More complex geometrics with plenty of details

2. Deep models are expected to be Scalable

3. Models are expected to be more accurate

Revisiting Transolver

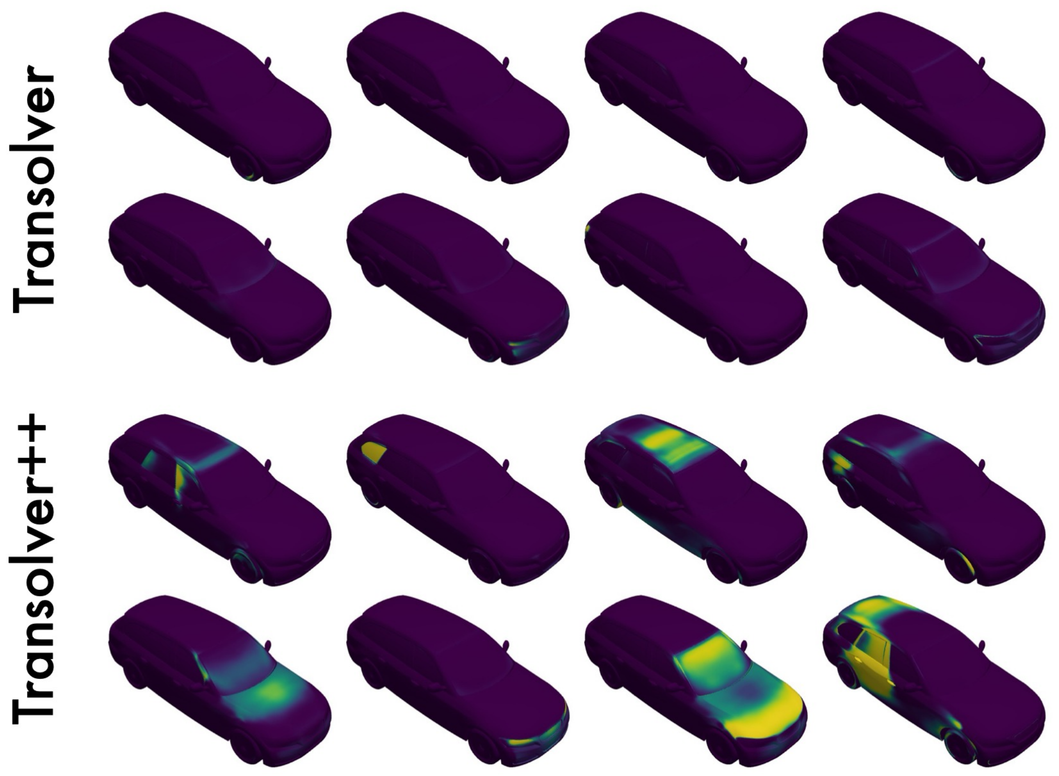


Transolver applies attention to learned physical states

① Mesh → physics ② Physics-Attention ③ Physics → Mesh

Challenges within Transolver in Million-Scale Geometries

1. Homogeneous physical states



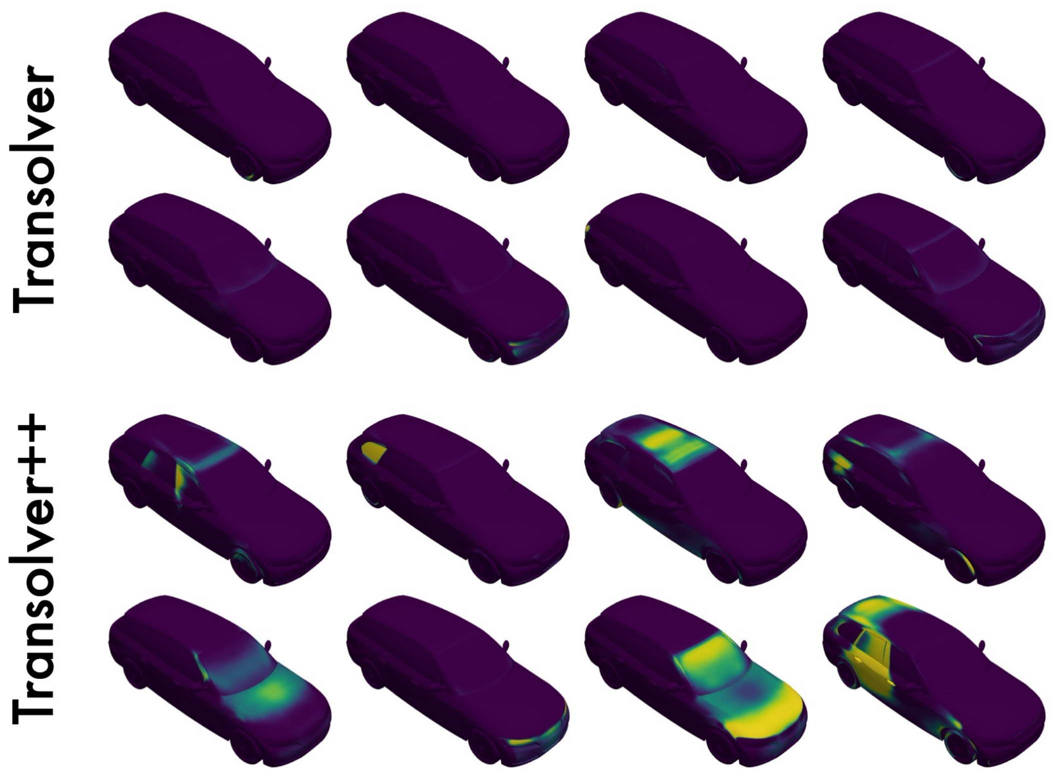
(b) Slice Weights Visualization

Degenerate in large-scale geometries

Improved physics learning

Challenges within Transolver in Million-Scale Geometries

1. Homogeneous physical states



(b) Slice Weights Visualization

2. Efficiency Bottleneck

$$\text{Slice weights: } \mathbf{w} = \text{Softmax}\left(\text{Linear}(\mathbf{x})/\tau_0\right)$$

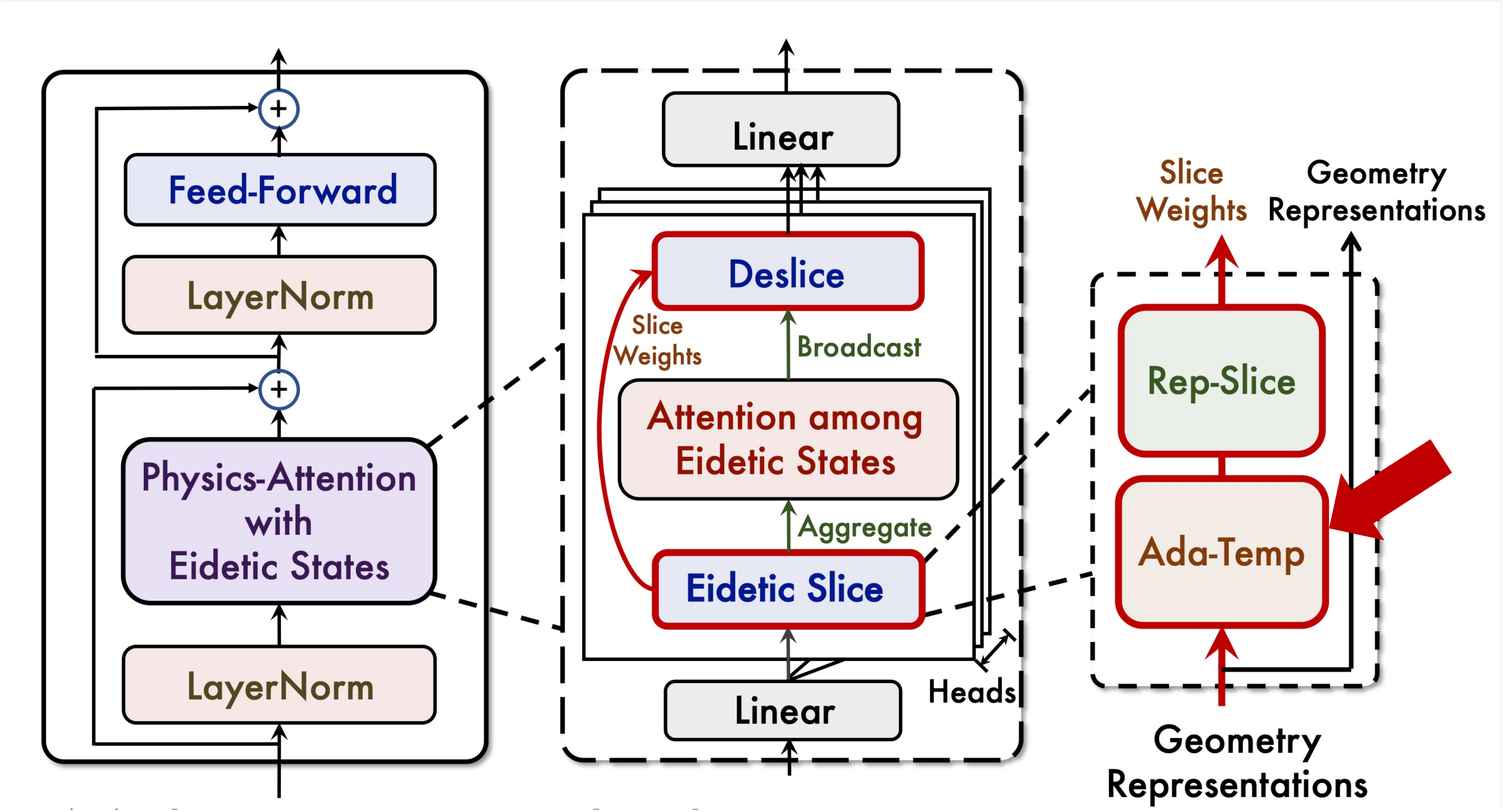
$$\text{Physical states: } \{\mathbf{s}_j\}_{j=1}^{M} = \left\{ \frac{\sum_{i=1}^{N} \mathbf{w}_{ij}\mathbf{x}_i}{\sum_{i=1}^{N} \mathbf{w}_{ij}} \right\}_{j=1}^{M}$$

- Even a single intermediate representation of one million mesh points will consume 2GB of GPU memory

- Previous upper bound of geometry scale is 600k on a single GPU supported by Transolver

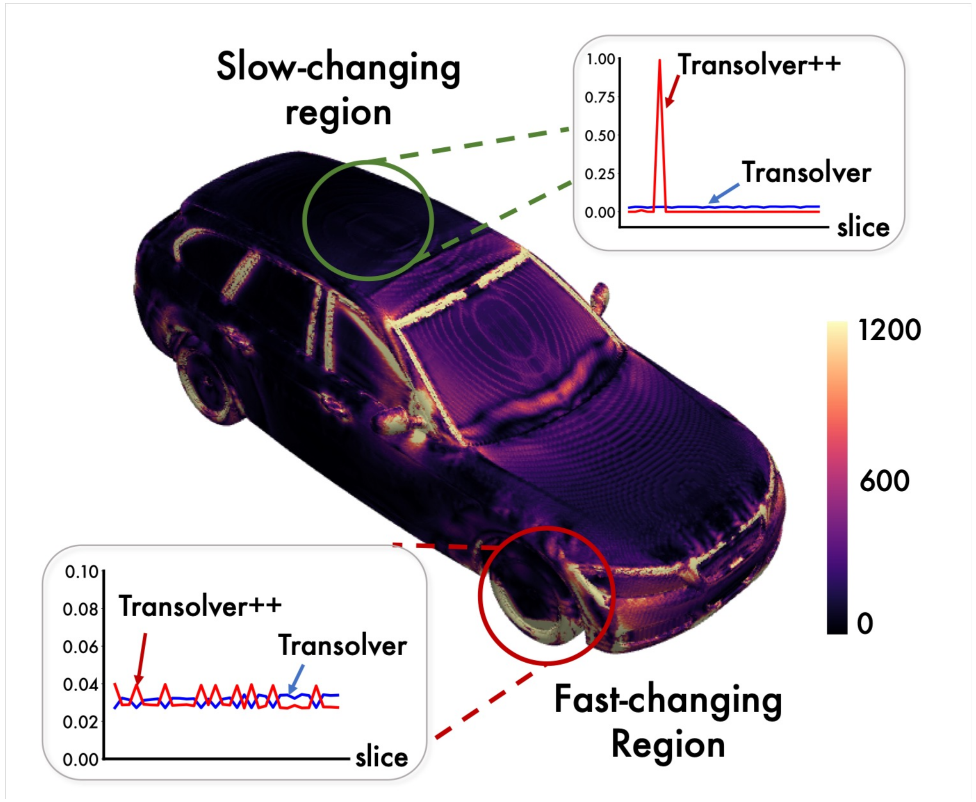
Upgrade 1: Physics-Attention with Eidetic States

Architectural Design



Upgrade 1: Physics-Attention with Eidetic States

Local Adaptive Mechanism



$$\text{Ada-Temp: } \tau = \{\tau_i\}_{i=1}^{N} = \{\tau_0 + \text{Linear}(\mathbf{x}_i)\}_{i=1}^{N},$$

- Utilize the local properties of each mesh point
- Learns the uncertainty of each points
- Adaptively change the temperature of each point

Slice reparameterization

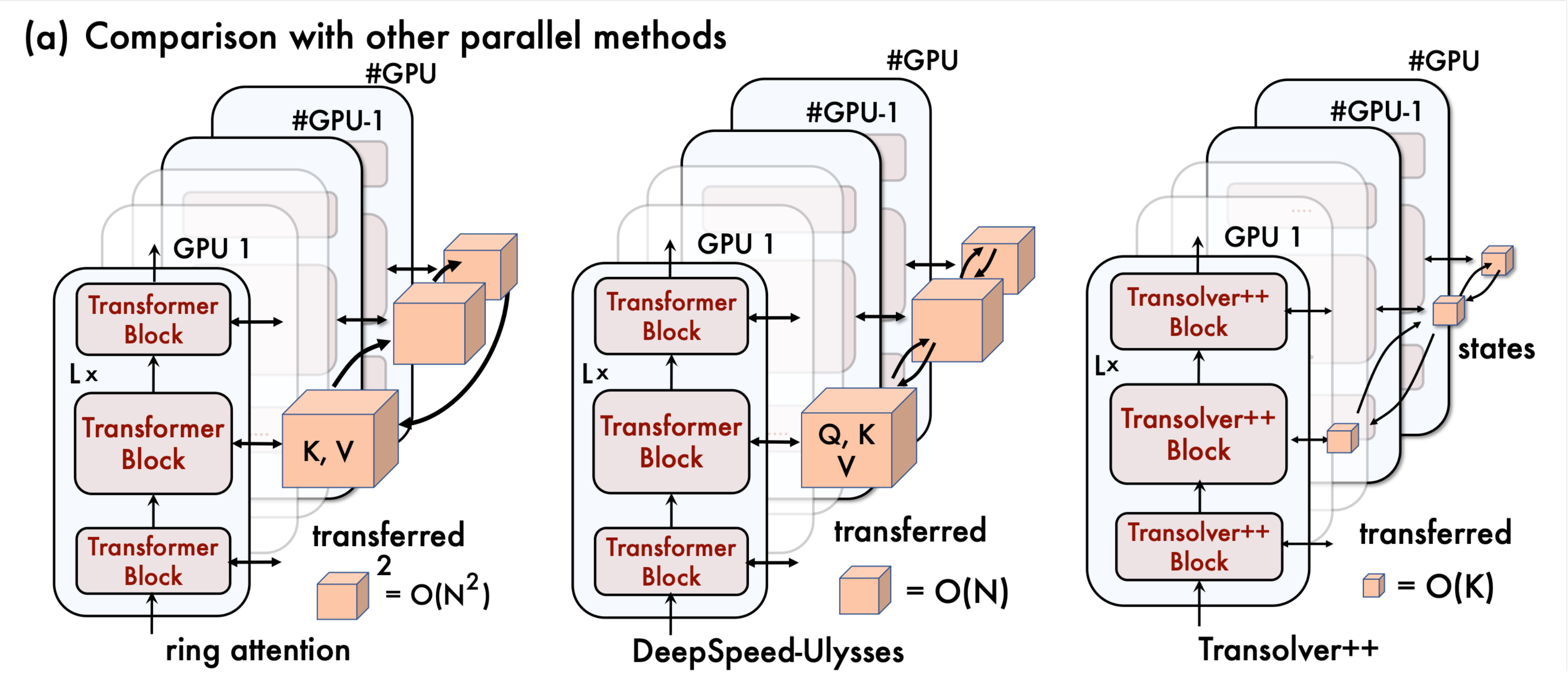
$$\text{Rep-Slice}(\mathbf{x}, \tau) = \text{Softmax}\left(\frac{\text{Linear}(\mathbf{x}) - \log(-\log \epsilon)}{\tau}\right), \quad (4)$$

Upgrade 2: Parallelism Framework

Transolver is under a natively parallel formulation.

Additivity of physical states:

$$\mathbf{s}_j = \frac{\sum_{i=1}^{N_1} \mathbf{w}_{ij}^{(1)} \mathbf{x}_i^{(1)} \oplus \cdots \oplus \sum_{i=1}^{N_{\#\text{gpu}}} \mathbf{w}_{ij}^{(\#\text{gpu})} \mathbf{x}_i^{(\#\text{gpu})}}{\sum_{i=1}^{N_1} \mathbf{w}_{ij}^{(1)} \oplus \cdots \oplus \sum_{i=1}^{N_{\#\text{gpu}}} \mathbf{w}_{ij}^{(\#\text{gpu})}}$$



Equivalent result

Accumulate **multi-GPU results**

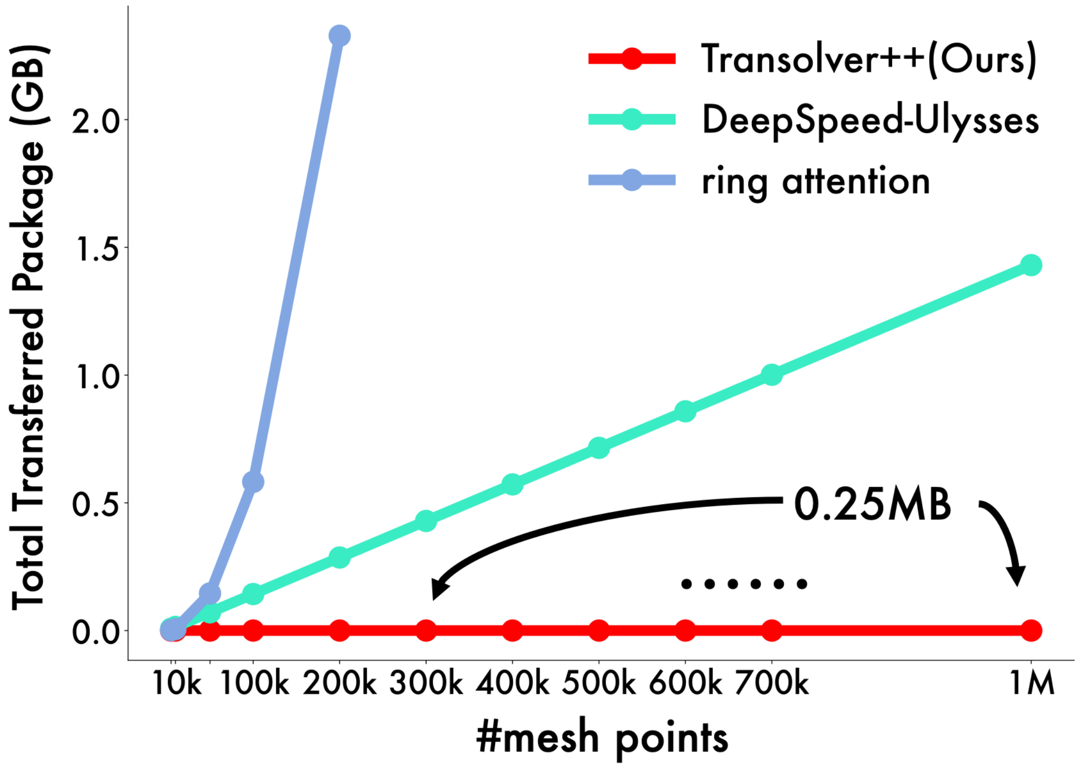
Compute physical states **in each GPU**

Split the mesh into **multiple GPUs**

Upgrade 2: Parallelism Framework

Overhead Analysis

(b) Scalability of Transferred Package

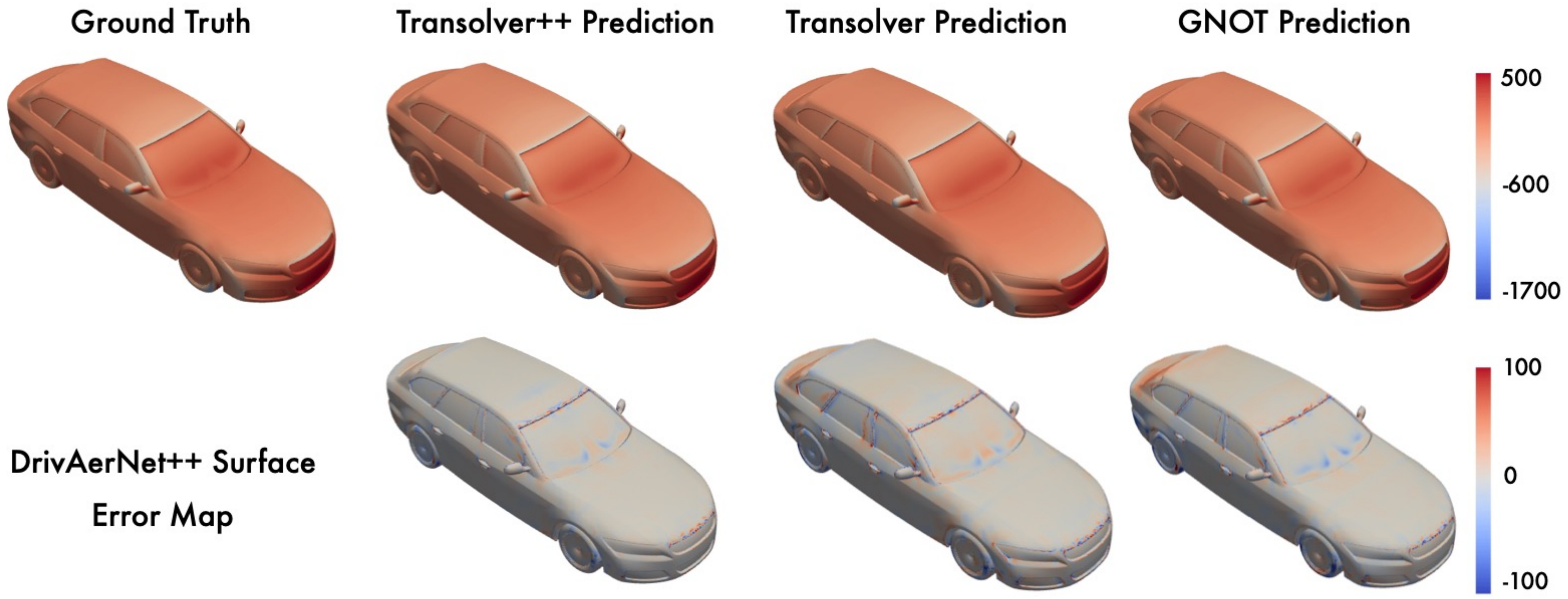


Further SpeedUp

Algorithm 1 Parallel Physics-Attention with Eidetic States

Input: Input features $\mathbf{x}^{(k)} \in \mathbb{R}^{N_k \times C}$ on the k-th GPU.
Output: Updated output features $\mathbf{x}'^{(k)} \in \mathbb{R}^{N_k \times C}$.
// drop f to save 50% memory.
Compute $\cancel{\mathbf{f}^{(k)}}, \mathbf{x}^{(k)} \leftarrow \text{Project}(\mathbf{x}^{(k)})$
Compute $\tau^{(k)} \leftarrow \tau_0 + \text{Ada-Temp}(\mathbf{x}^{(k)})$
Compute weights $\mathbf{w}^{(k)} \leftarrow \text{Rep-Slice}(\mathbf{x}^{(k)}, \tau^{(k)})$
Compute weights norm $\mathbf{w}_{\text{norm}}^{(k)} \leftarrow \sum_{i=1}^{N_k} \mathbf{w}_i^{(k)}$
Reduce slice norm $\mathbf{w}_{\text{norm}} \leftarrow \text{AllReduce}(\mathbf{w}_{\text{norm}}^{(k)})$ $\mathcal{O}(M)$
Compute eidetic states $\mathbf{s}^{(k)} \leftarrow \frac{\mathbf{w}^{(k)\top}\mathbf{x}^{(k)}\cancel{\mathbf{f}^{(k)}}}{\mathbf{w}_{\text{norm}}}$
Reduce eidetic states $\mathbf{s} \leftarrow \text{AllReduce}(\mathbf{s}^{(k)})$ $\mathcal{O}(MC)$
Update eidetic states $\mathbf{s}' \leftarrow \text{Attention}(\mathbf{s})$
Deslice back to $\mathbf{x}'^{(k)} \leftarrow \text{Deslice}(\mathbf{s}', \mathbf{w}^{(k)})$
Return $\mathbf{x}'^{(k)}$

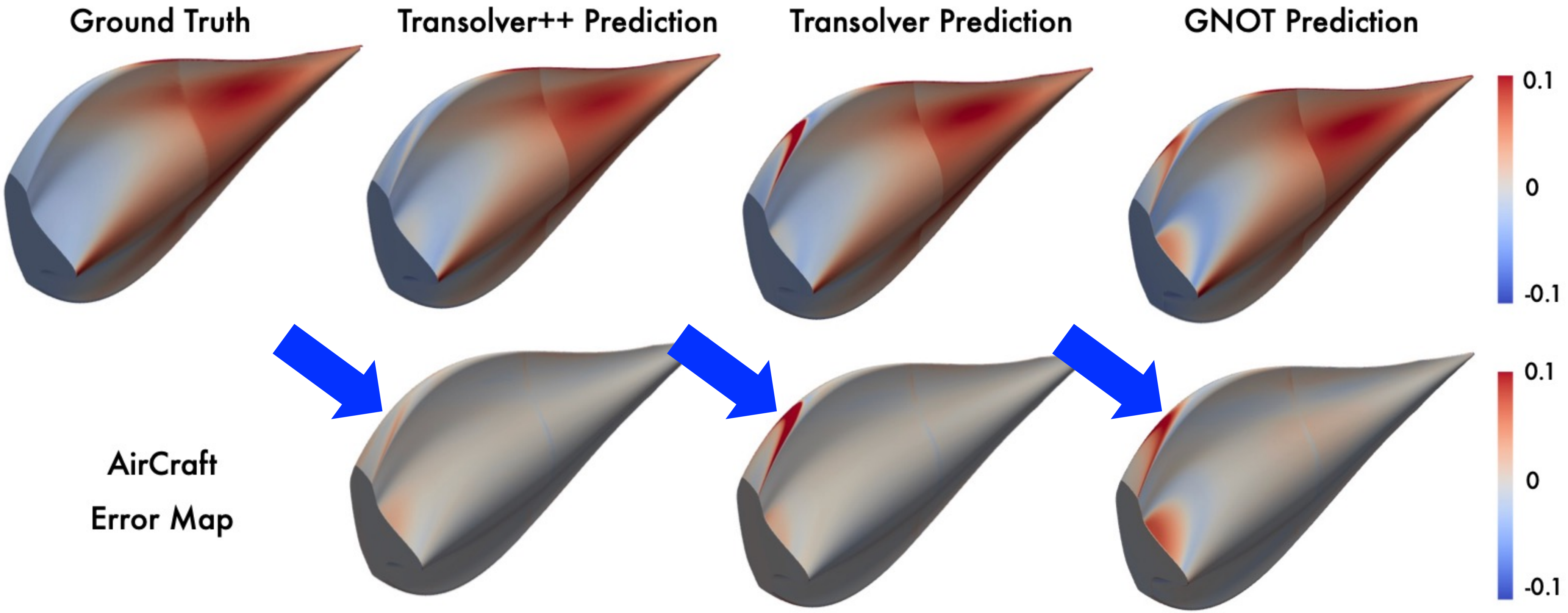
Industrial-level Applications: Car Design



Transolver++ achieves over 20% error reduction than other models.

Relative Drag Coefficient Error = 3.6%; Relative Field Error = 11%.

Industrial-level Applications: AirCraft Design



Transolver++ achieves over 20% error reduction than other models.

Relative Drag Coefficient Error = 1.4%; Relative Field Error = 6.4%.

Back to Transolver's Original Design!

Transolver-3: Scaling Up Transformer Solvers to Industrial-Scale Geometries

Hang Zhou [1] **Haixu Wu** [1] **Haonan Shangguan** [1] **Yuezhou Ma** [1] **Huikun Weng** [1] **Jianmin Wang** [1] **Mingsheng Long** [1]


Hang Zhou


Haixu Wu


Haonan ShangGuan

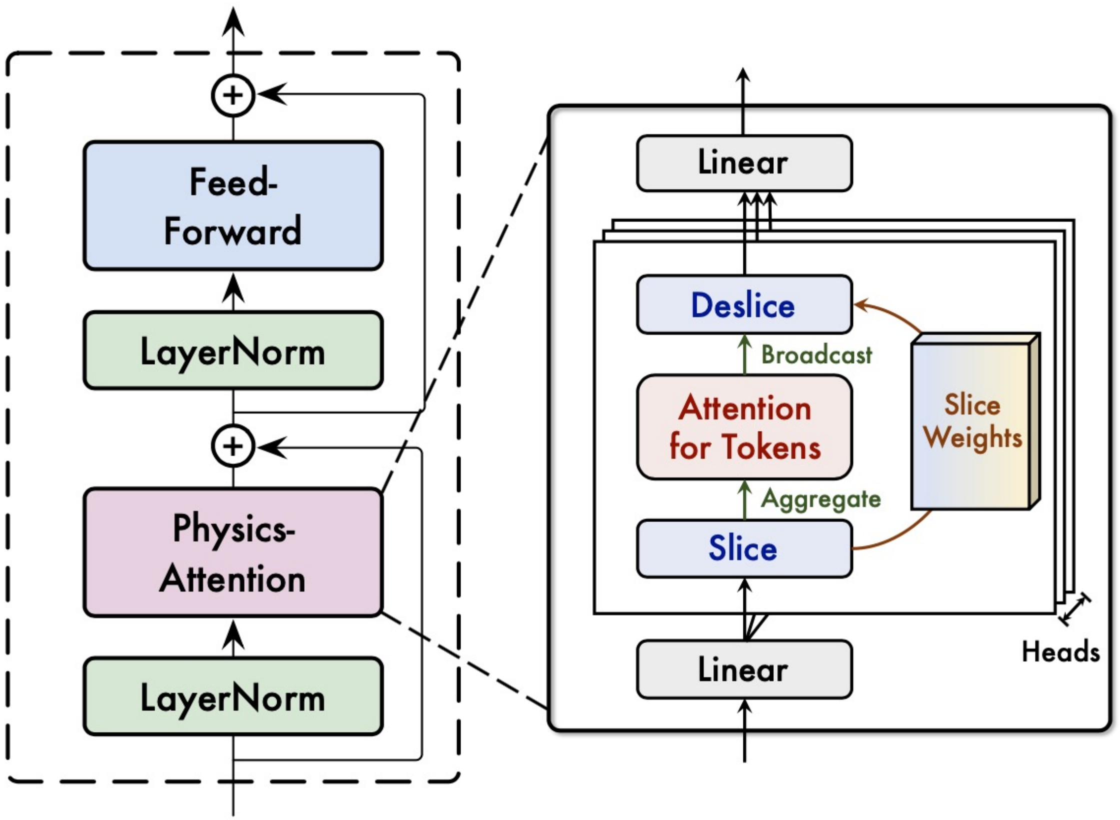

Yuezhou Ma

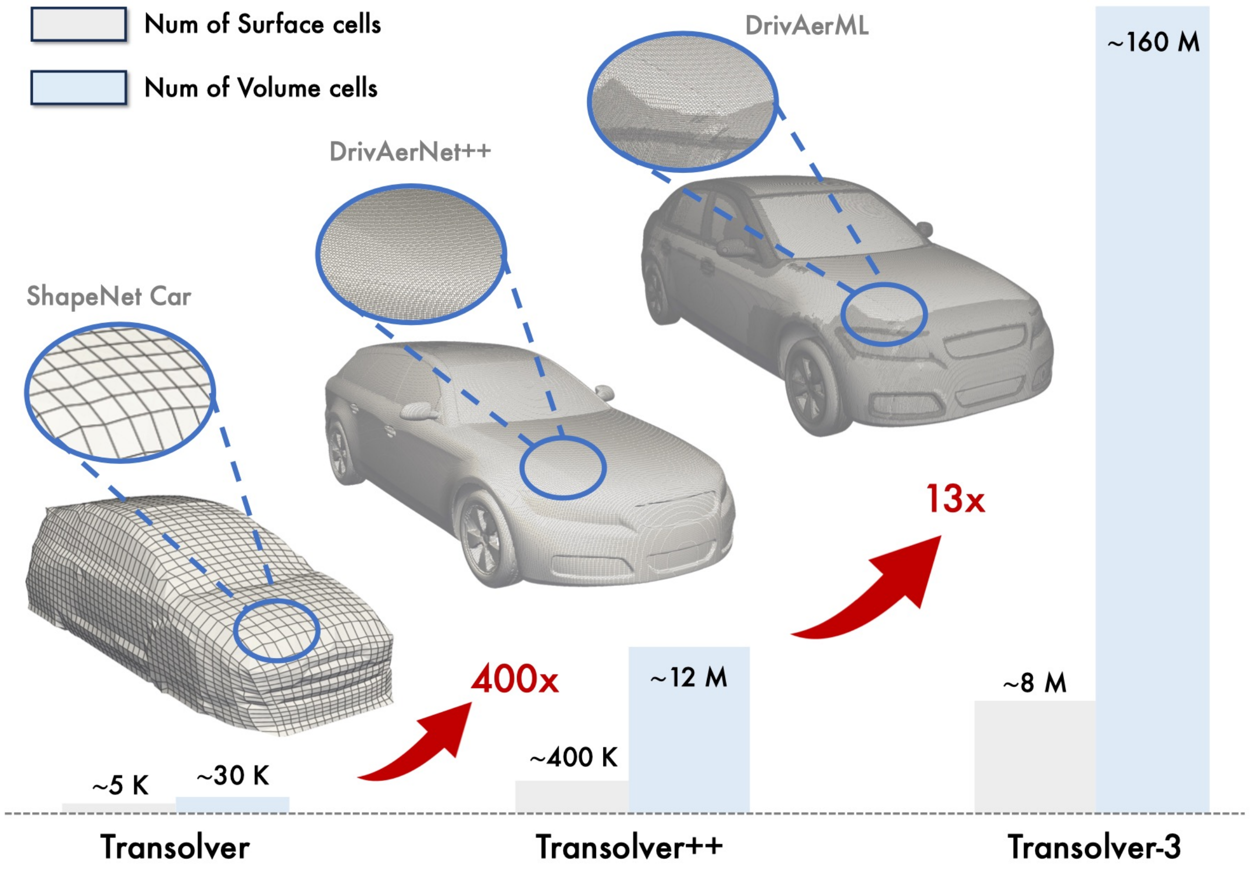

Huikun Weng


Jianmin Wang


Mingsheng Long

Scale to Over 100-Million-Cell Geometries





Detailed Complexity Analysis

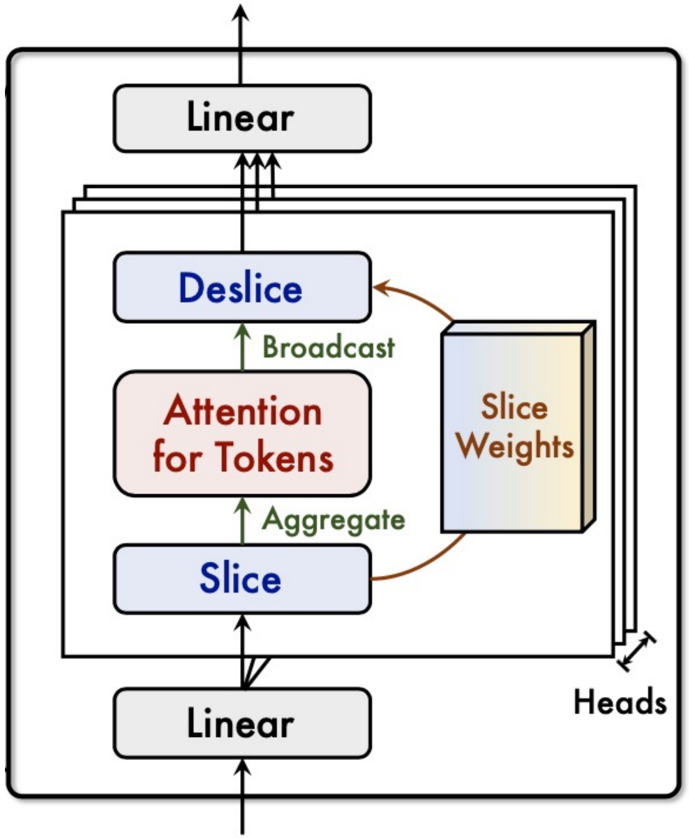


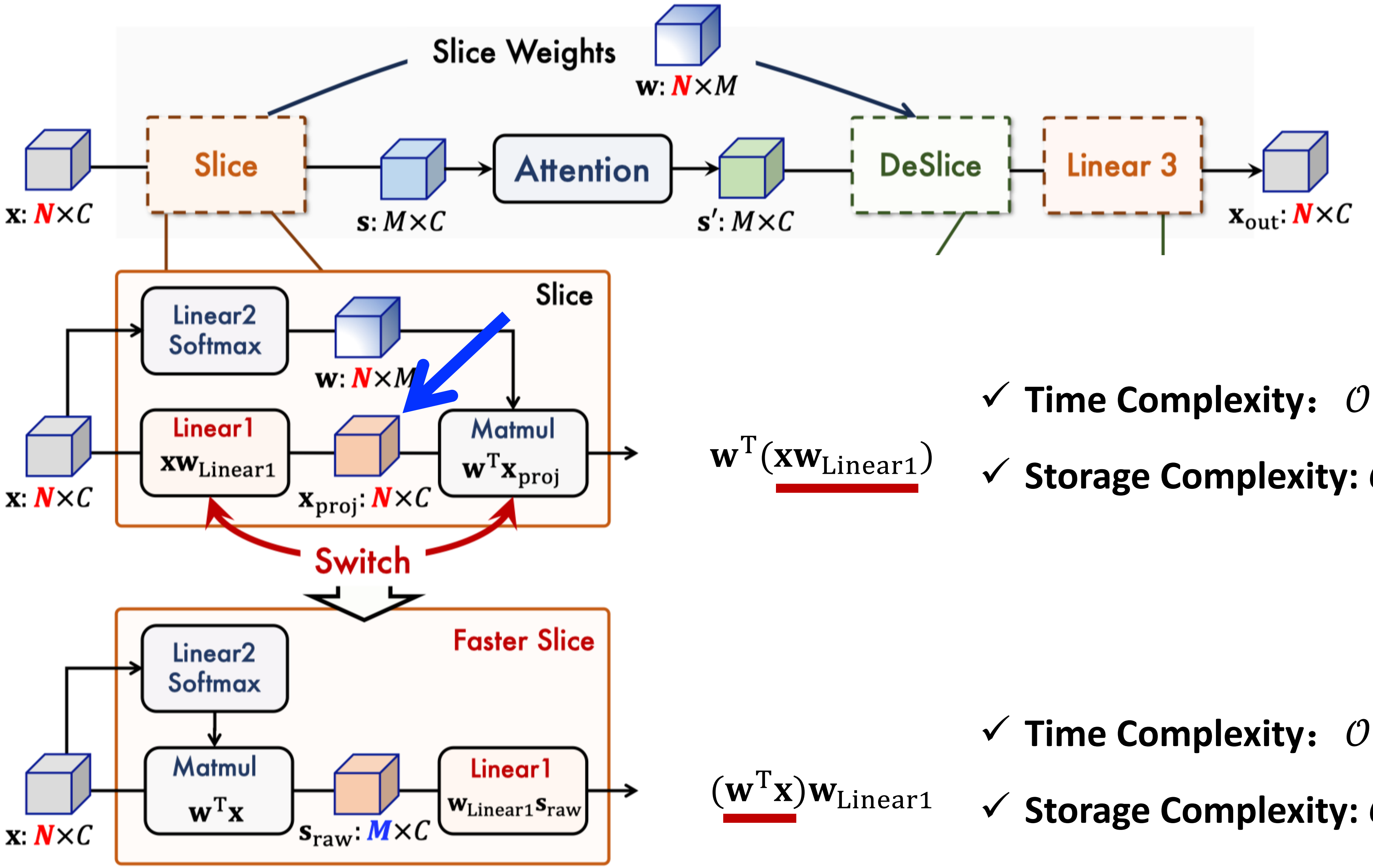
Table 1. Complexity Analysis of Original Physics-Attention.

Operation	Time Complexity	Space Complexity	
Linear1($\mathbf{x}$)	$O(NC^2)$	$O(NC)$	Slice
Softmax(Linear2($\mathbf{x}$))	$O(NCM)$	$O(NM)$	
$(\mathbf{wd}^{-1})^\top \mathbf{x}_{\text{proj}}$	$O(NMC)$	$O(MC)$	
Attention($\mathbf{s}$)	$O(M^2C)$	$O(M^2 + MC)$	Attn
$\mathbf{ws}'$	$O(NMC)$	$O(NC)$	Deslice
Linear3($\mathbf{ws}'$)	$O(NC^2)$	$O(NC)$	
N-Related Terms	5	4	

N (mesh size) >> C (hidden channels) >= M (physical states)

we should care about all the terms related to *N*.

Faster Slice



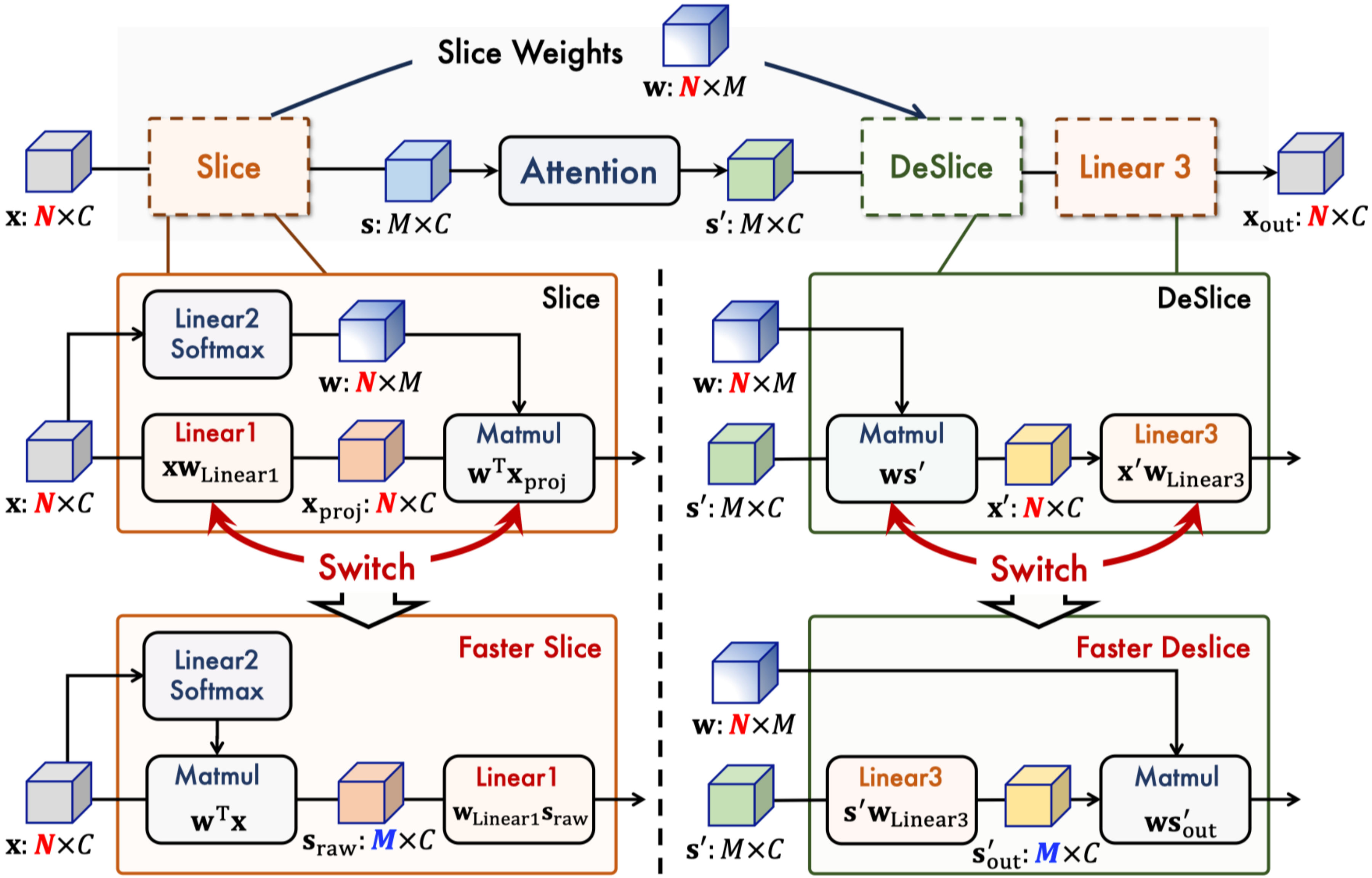
$\mathbf{w}^{\mathrm{T}}(\mathbf{x}\mathbf{w}_{\mathrm{Linear1}})$

- ✓ **Time Complexity:** $\mathcal{O}(NC^2 + NMC)$
- ✓ **Storage Complexity:** $\mathcal{O}(NM + NC)$

$(\mathbf{w}^{\mathrm{T}}\mathbf{x})\mathbf{w}_{\mathrm{Linear1}}$

- ✓ **Time Complexity:** $\mathcal{O}(MC^2 + NMC)$
- ✓ **Storage Complexity:** $\mathcal{O}(NM + MC)$

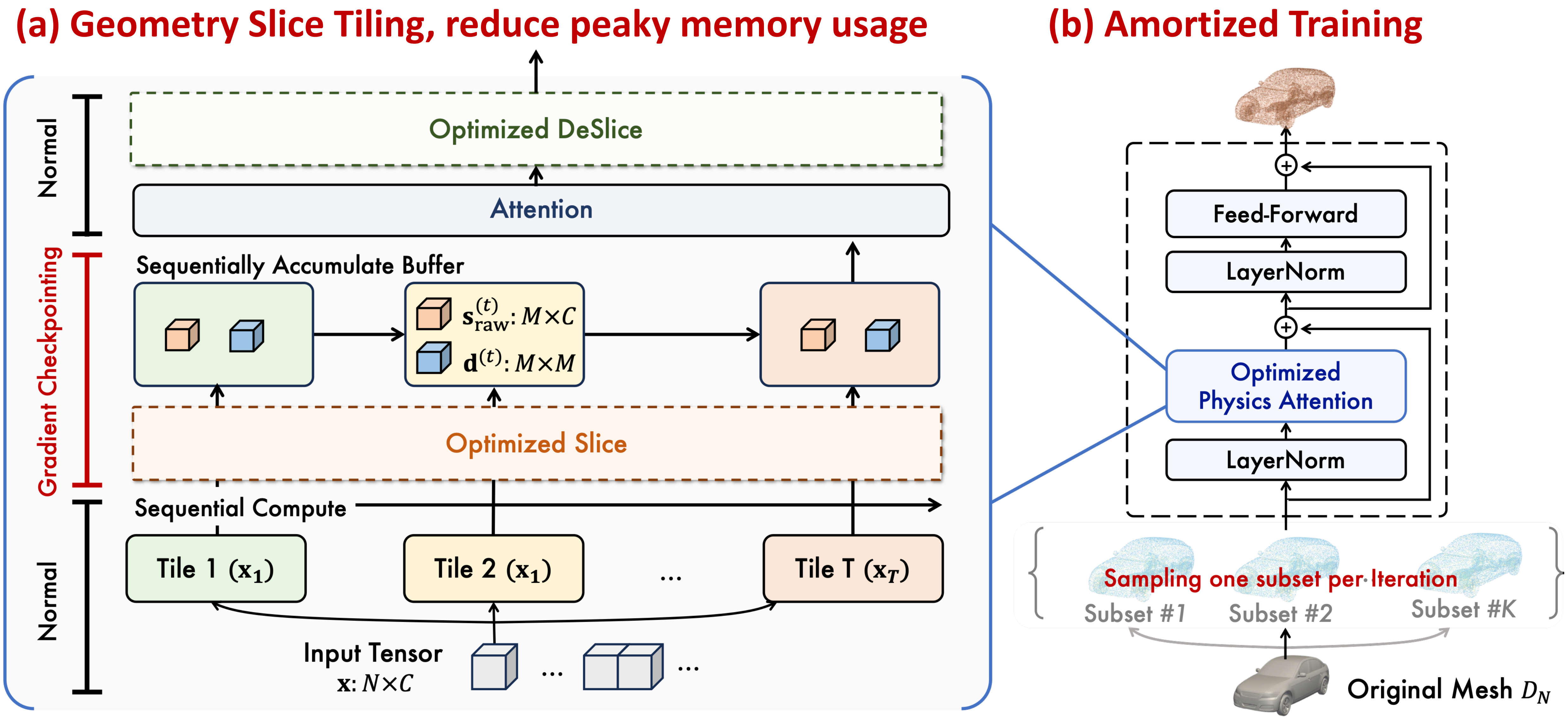
Faster DeSlice



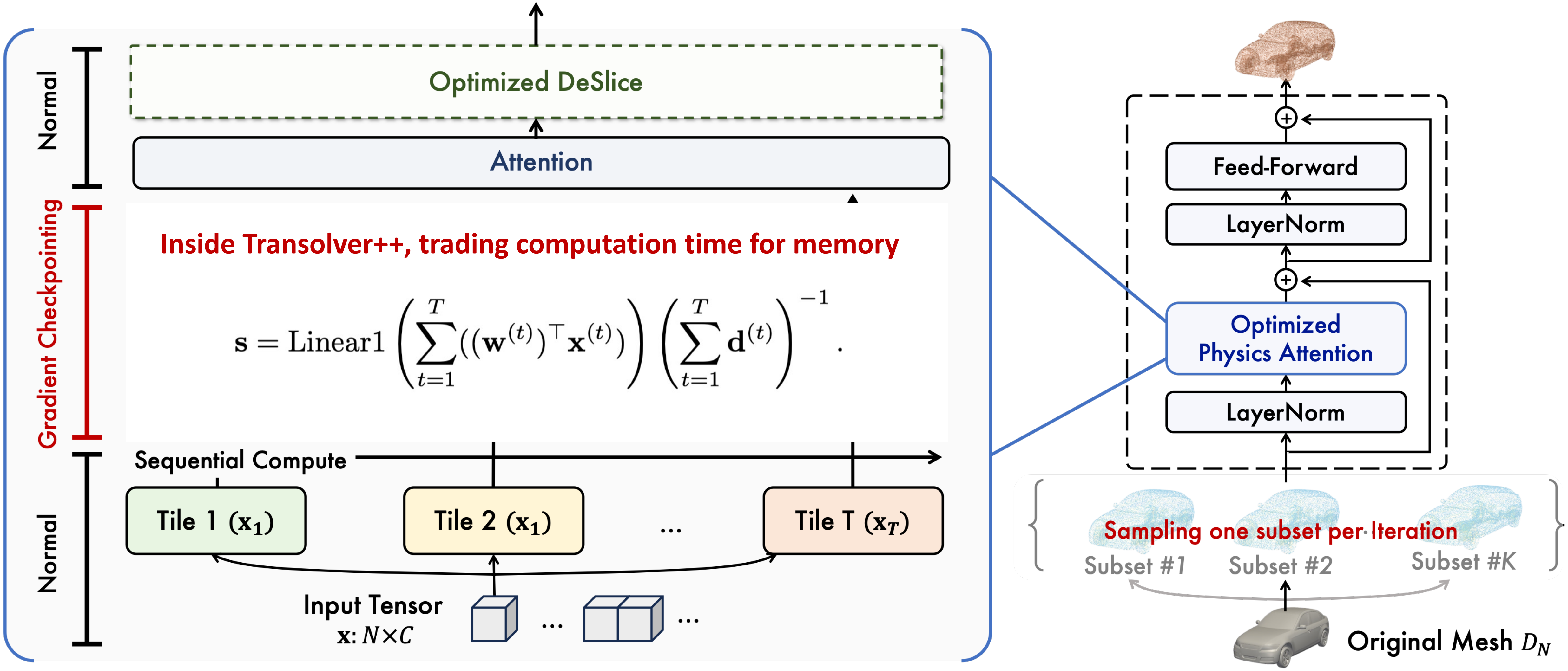
$(\mathbf{ws'})\mathbf{w}_{\text{Linear3}}$

$\mathbf{w}(\mathbf{s'}\mathbf{w}_{\text{Linear3}})$

Training Scaling Framework



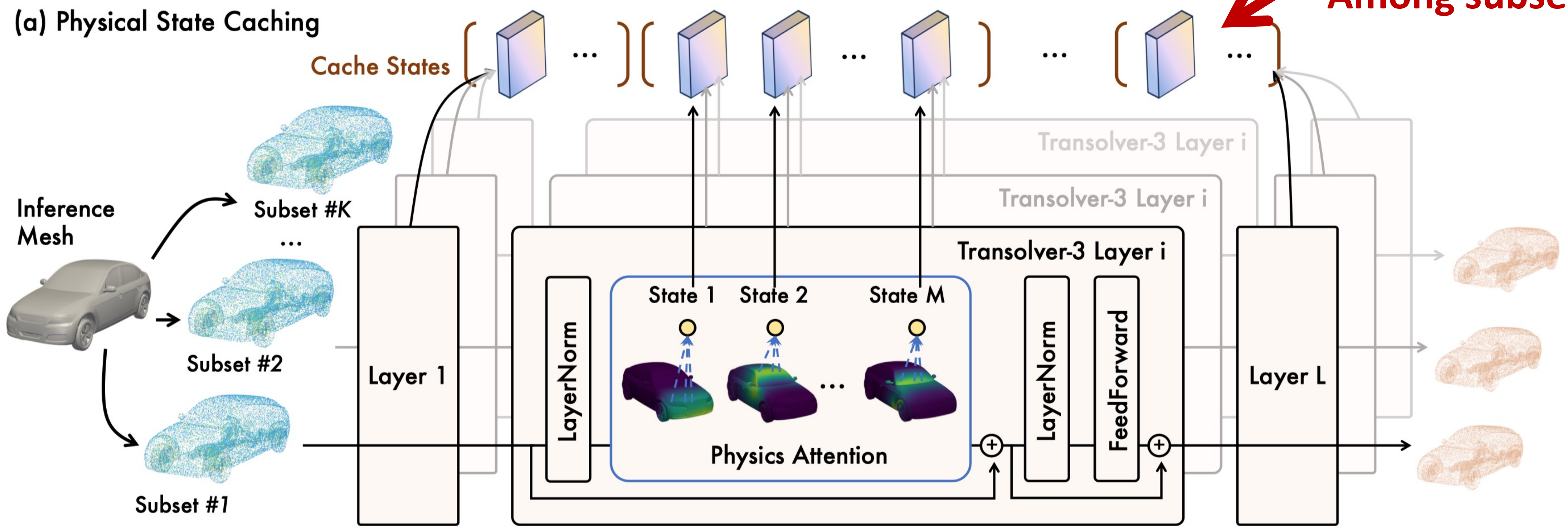
Training Scaling Framework



Inference Scaling Framework

Amortized training separates the PDE solving process into several subsets, successfully reducing memory, but it **cannot get the correct physical state**.

Accumulate
Among subsets



Inference Scaling Framework

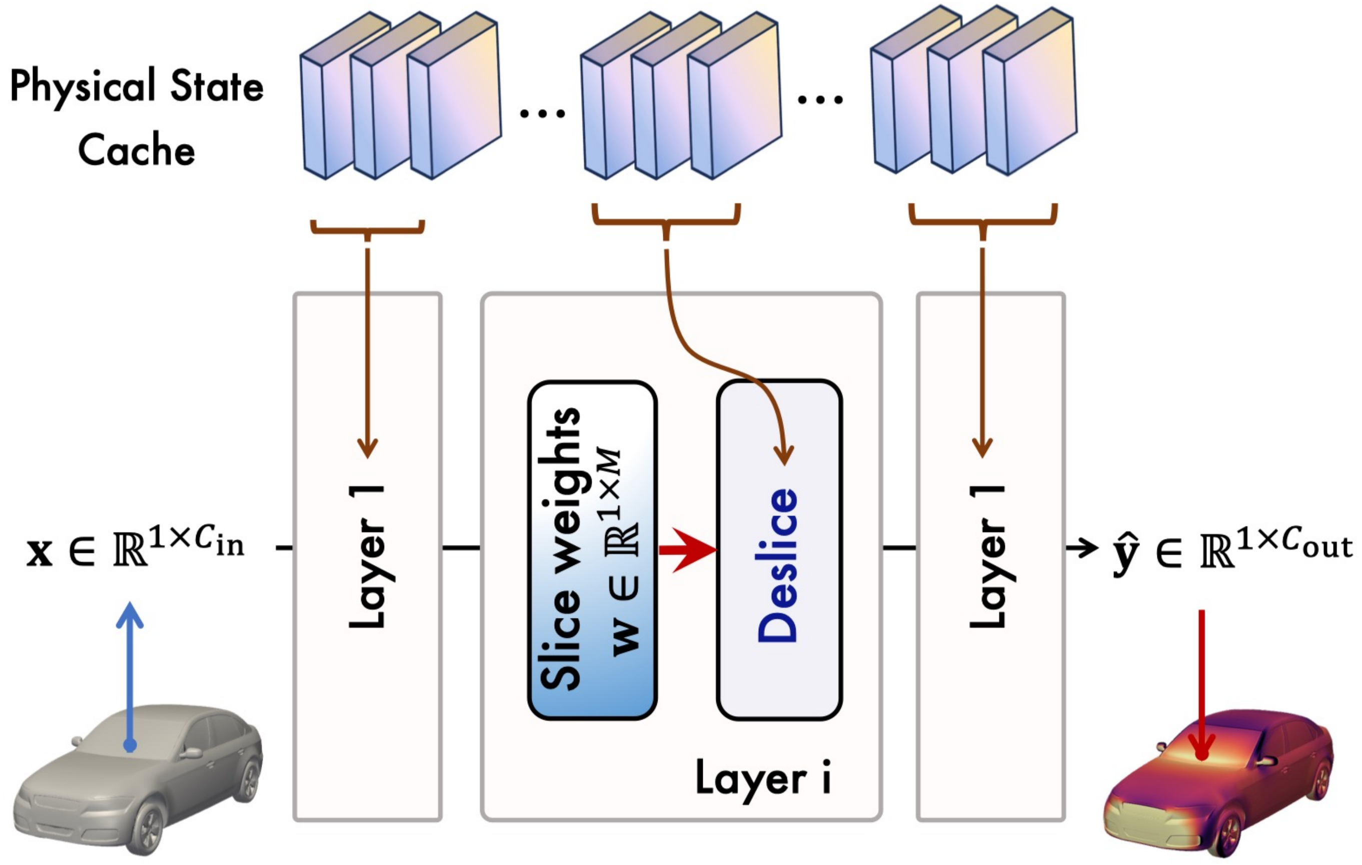
Inference on the **arbitrary position (in PINN style).**

$$\mathbf{w}^{(l)} = \mathrm{Softmax}\left(\mathrm{Linear2}(\mathbf{x}^{(l)})\right)$$

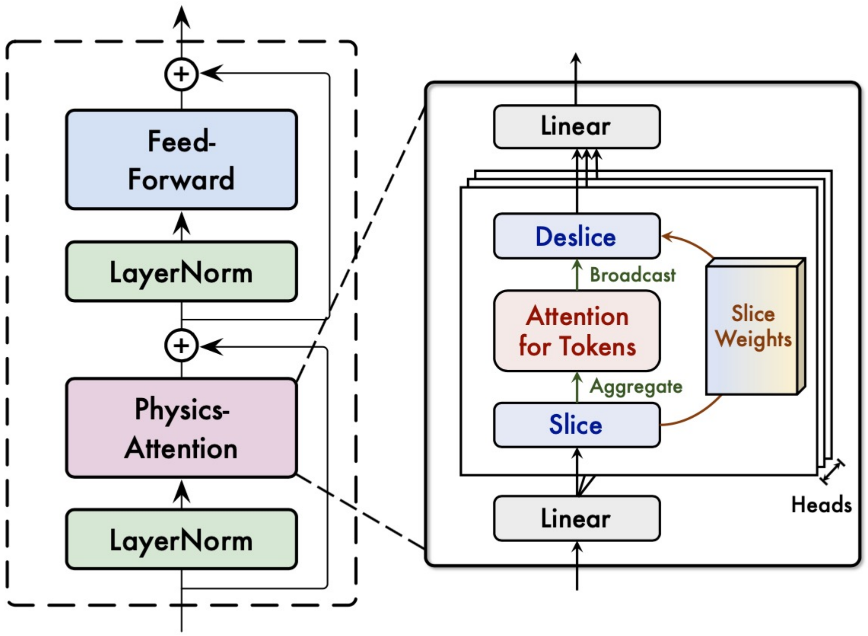
$$\mathbf{x}_{\mathbf{out}}^{(l)} = \mathbf{w}^{(l)}\mathbf{s}'^{(l)}_{\mathrm{out}}$$

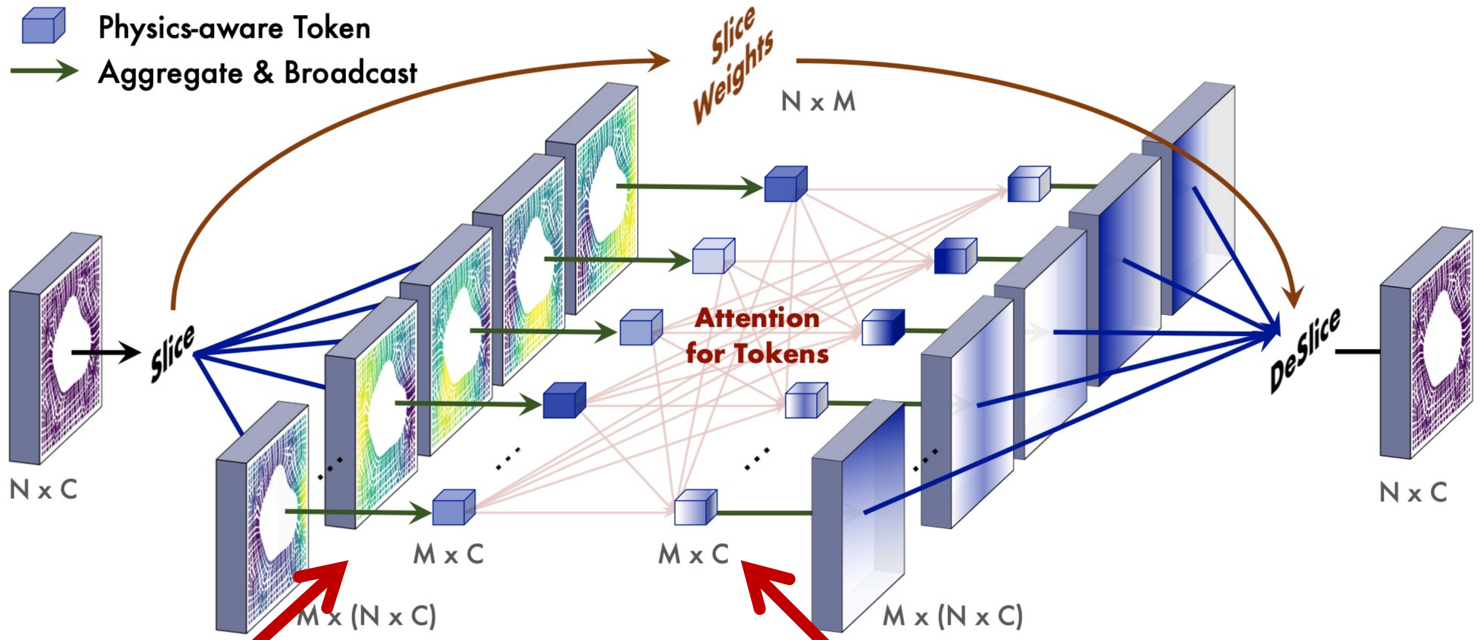
Cached physical states ($\mathbf{s}'^{(l)}_{\mathrm{out}}$)

Newly estimated slice weights ($\mathbf{w}^{(l)}$)



"Magic Design" in Transolver





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

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

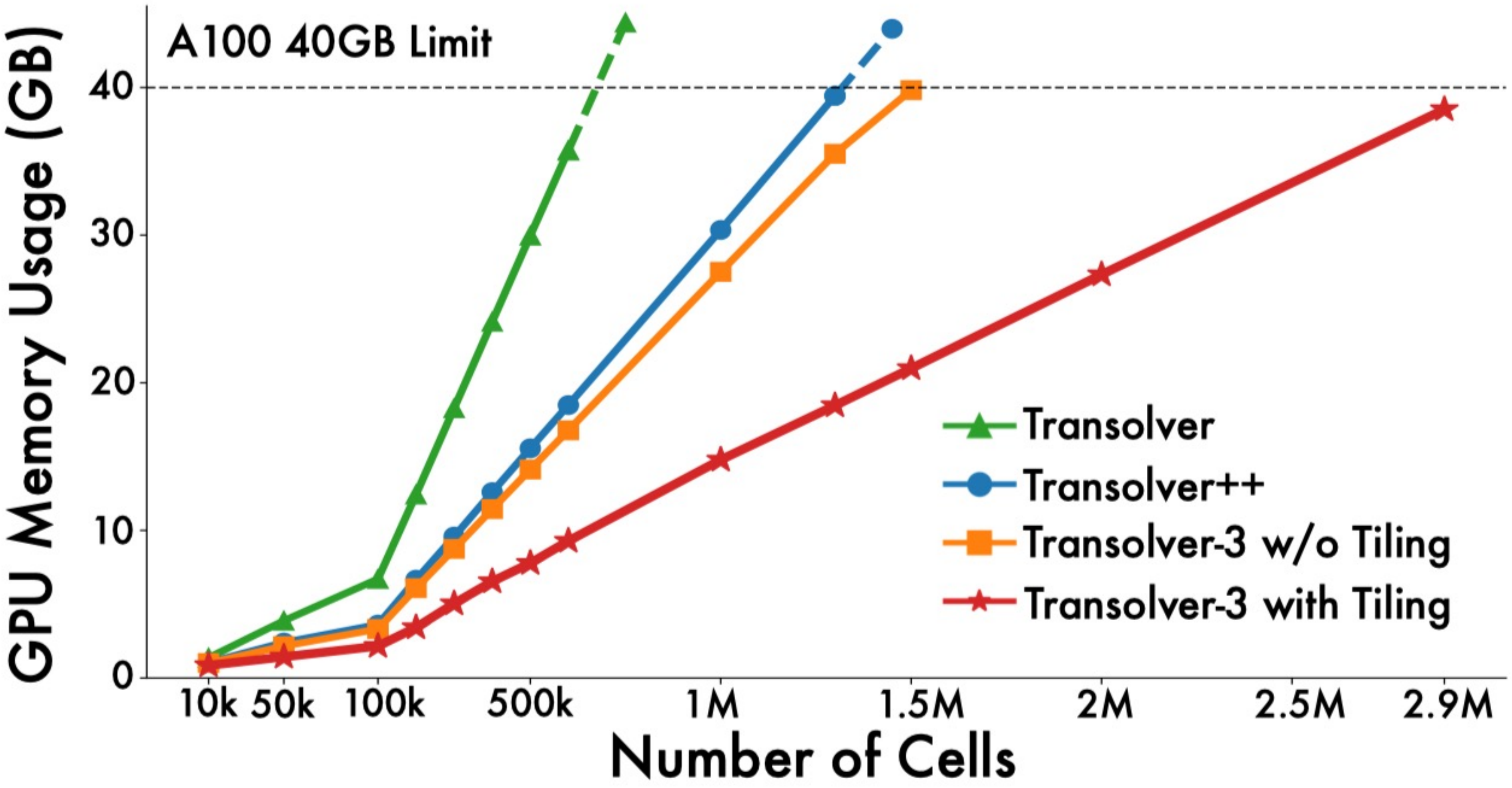
Why adopt the global weighted sum?

Support Transolver++

Why reuse slice weights?

Support Transolver-3

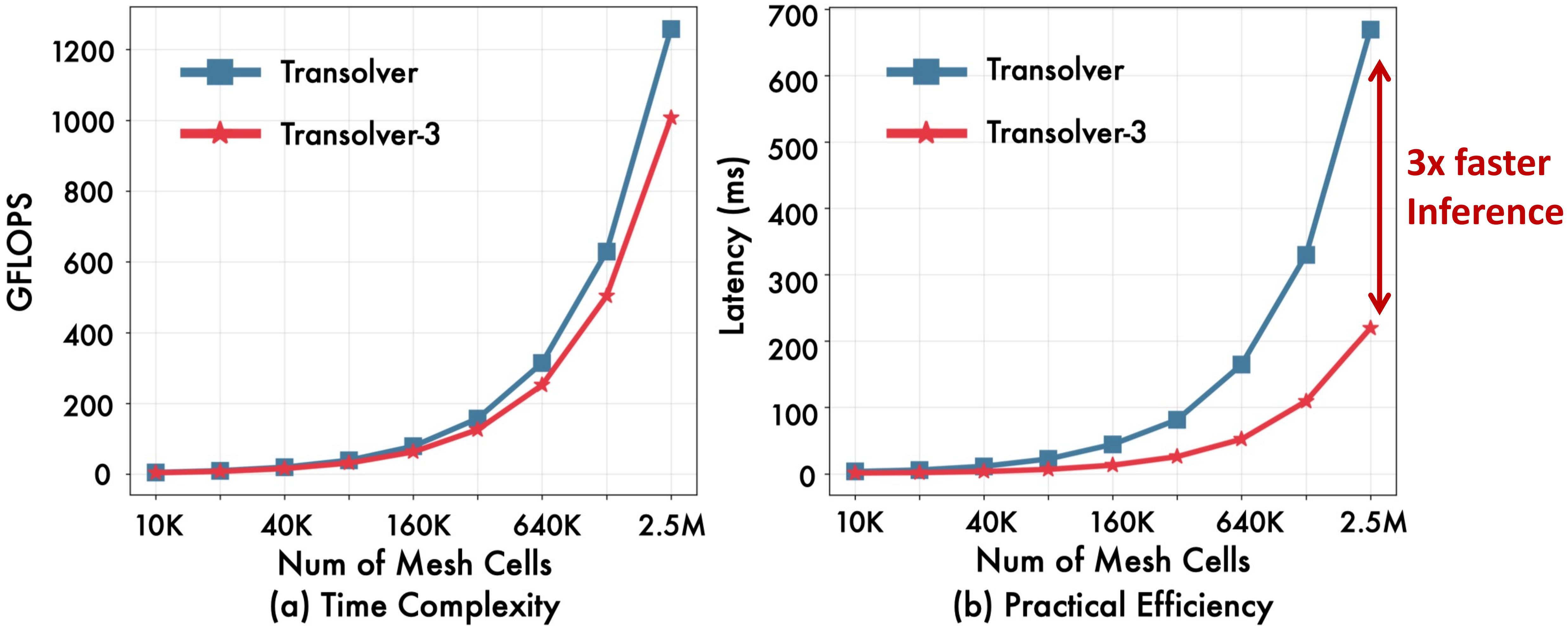
Efficiency Analysis (Geometry Scaling)



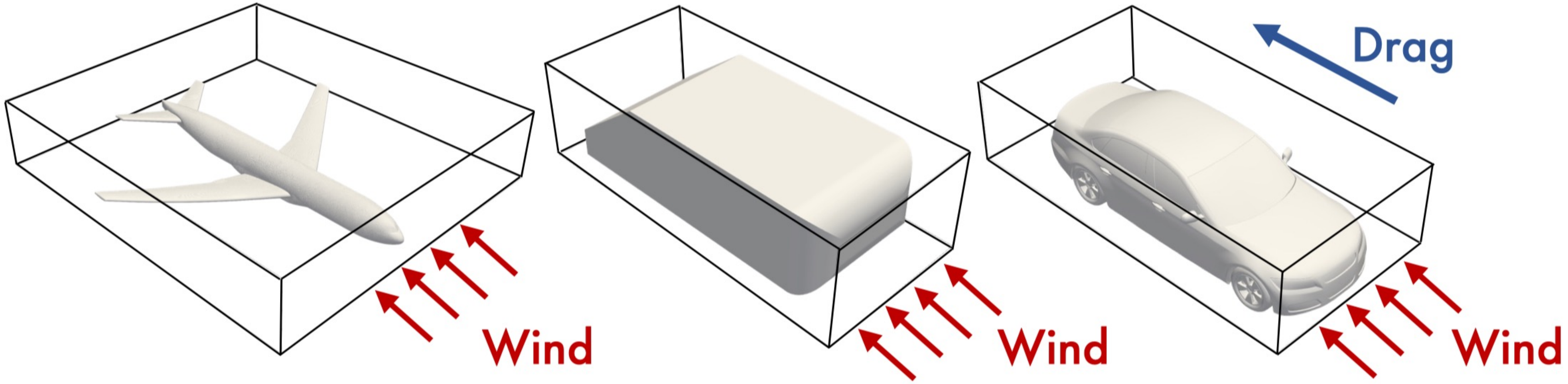
With slice tiling, Transolver-3 can process around 3M points on a single GPU.

5x larger than vanilla Transolver, 2x larger than Transolver++

Efficiency Analysis (Inference Latency)



Experiments



(a) NASA-CRM (b) AhmedML (c) DrivAerML

400K cells per sample	20M cells per sample	160M cells per sample
4 GB	8 TB	31 TB

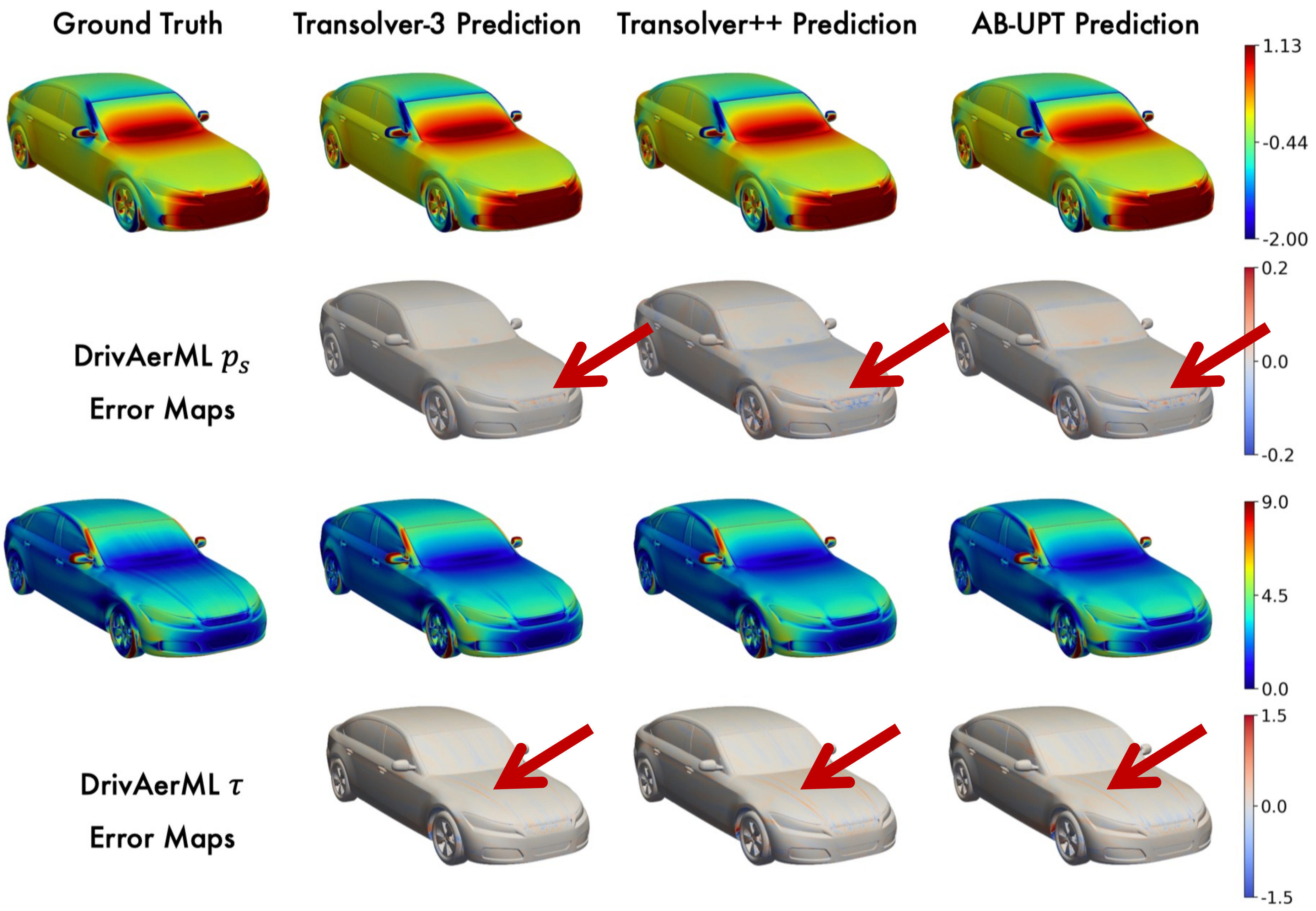
Main Results

Table 4. Relative L2 errors (in %) of surface pressure $\boldsymbol{p}_s$ and skin friction coefficient $\boldsymbol{C}_f$ on the NASA-CRM dataset, and surface pressure $\boldsymbol{p}_s$, volume velocity $\boldsymbol{u}$, wall shear stress $\boldsymbol{\tau}$ and volume pressure $\boldsymbol{p}_v$ on the AhmedML and DrivAerML datasets.

MODELS	NASA-CRM		AHMEDML				DRIVAERML			
	$\boldsymbol{p}_s$	$\boldsymbol{C}_f$	$\boldsymbol{p}_s$	$\boldsymbol{u}$	$\boldsymbol{\tau}$	$\boldsymbol{p}_v$	$\boldsymbol{p}_s$	$\boldsymbol{u}$	$\boldsymbol{\tau}$	$\boldsymbol{p}_v$
GRAPH U-NET*	15.85	15.61	6.46	4.15	7.29	5.18	16.13	17.98	27.84	20.51
GINO*	12.39	11.51	7.90	6.23	8.18	8.80	13.03	40.58	21.71	44.90
GAOT*	30.38	59.79	8.02	7.43	9.92	10.47	34.00	57.18	61.00	56.90
UPT	12.78	23.78	4.25	2.73	5.80	3.10	7.44	8.74	12.93	10.05
AB-UPT	9.77	<u>6.43</u>	3.97	1.94	5.60	**2.07**	<u>3.82</u>	5.93	7.29	<u>6.08</u>
TRANSOLVER*	9.61	7.04	<u>3.20</u>	1.81	<u>4.85</u>	2.41	4.81	6.78	8.95	7.74
TRANSOLVER++*	<u>9.51</u>	6.95	3.47	<u>1.78</u>	5.06	2.35	4.12	<u>4.70</u>	<u>6.42</u>	6.70
TRANSOLVER-3	**8.71**	**5.85**	**2.96**	**1.60**	**4.81**	<u>2.16</u>	**3.71**	**4.14**	**5.85**	**5.72**

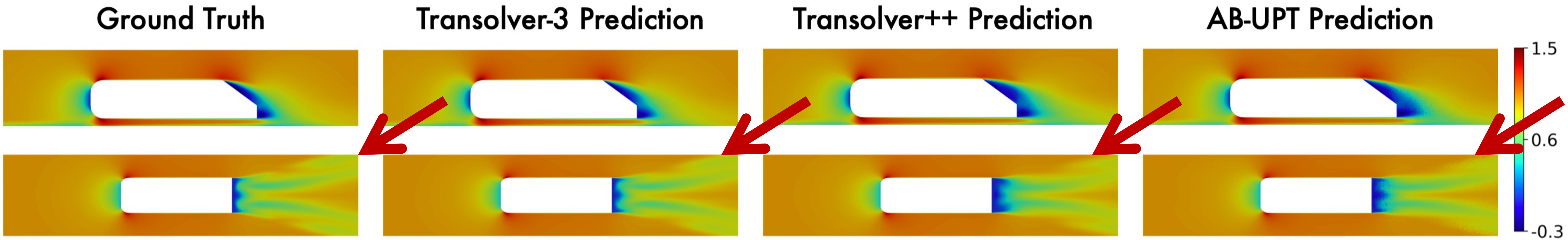
Without any architecture change, only upgrade training and inference paradigms.

Transolver still achieves the best performance.

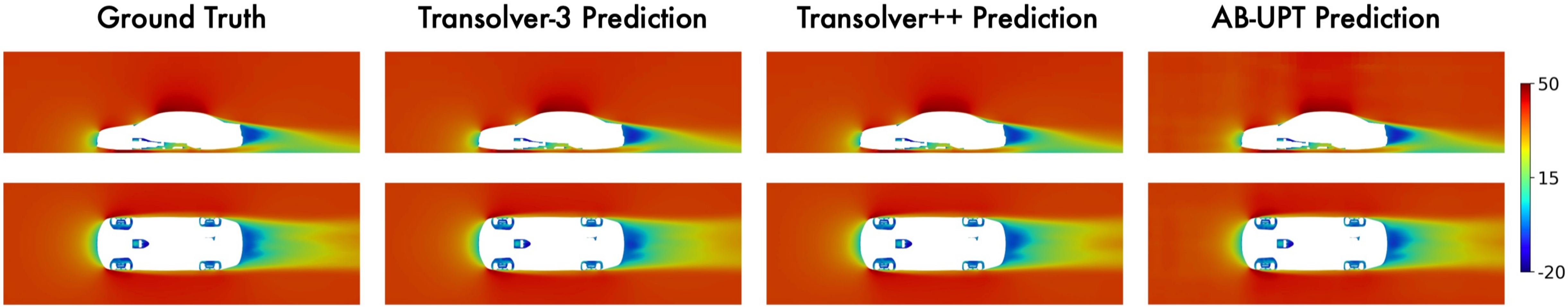

Ground Truth
Transolver-3 Prediction
Transolver++ Prediction
AB-UPT Prediction
DrivAerML p_s
Error Maps
DrivAerML τ
Error Maps
1.13
-0.44
-2.00
0.2
0.0
-0.2
9.0
4.5
0.0
1.5
0.0
-1.5

Showcase study

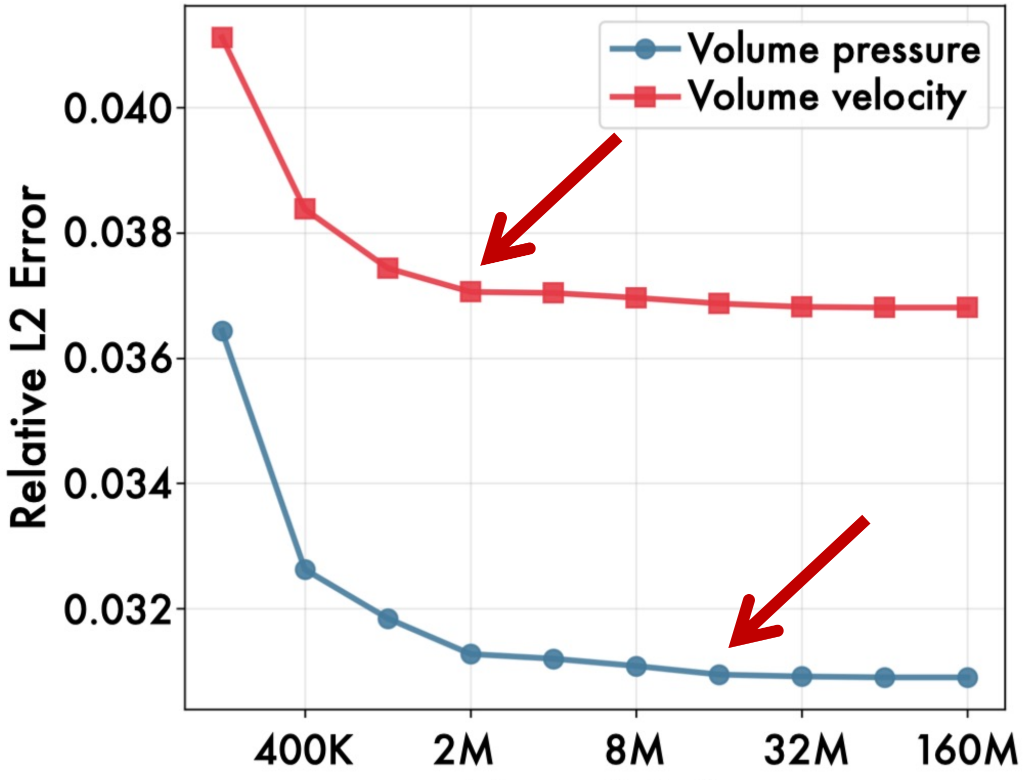
(1) AhmedML Benchmark



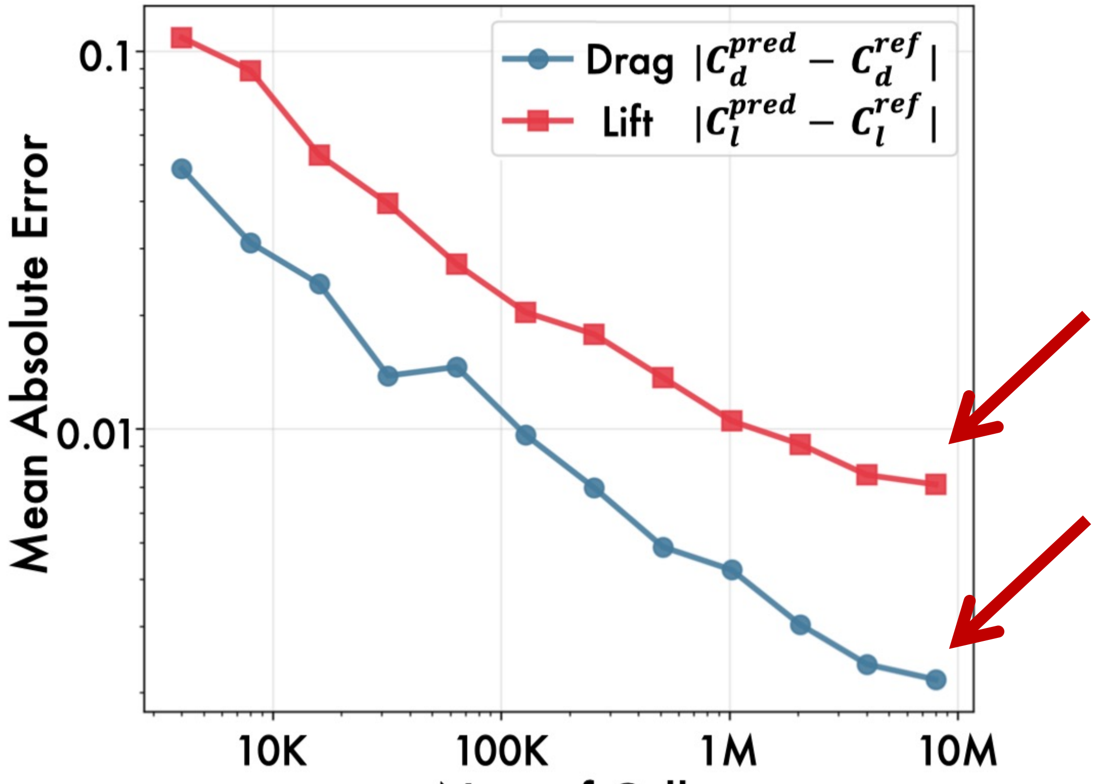
(2) DrivAerML Benchmark



Why Geometry Scaling

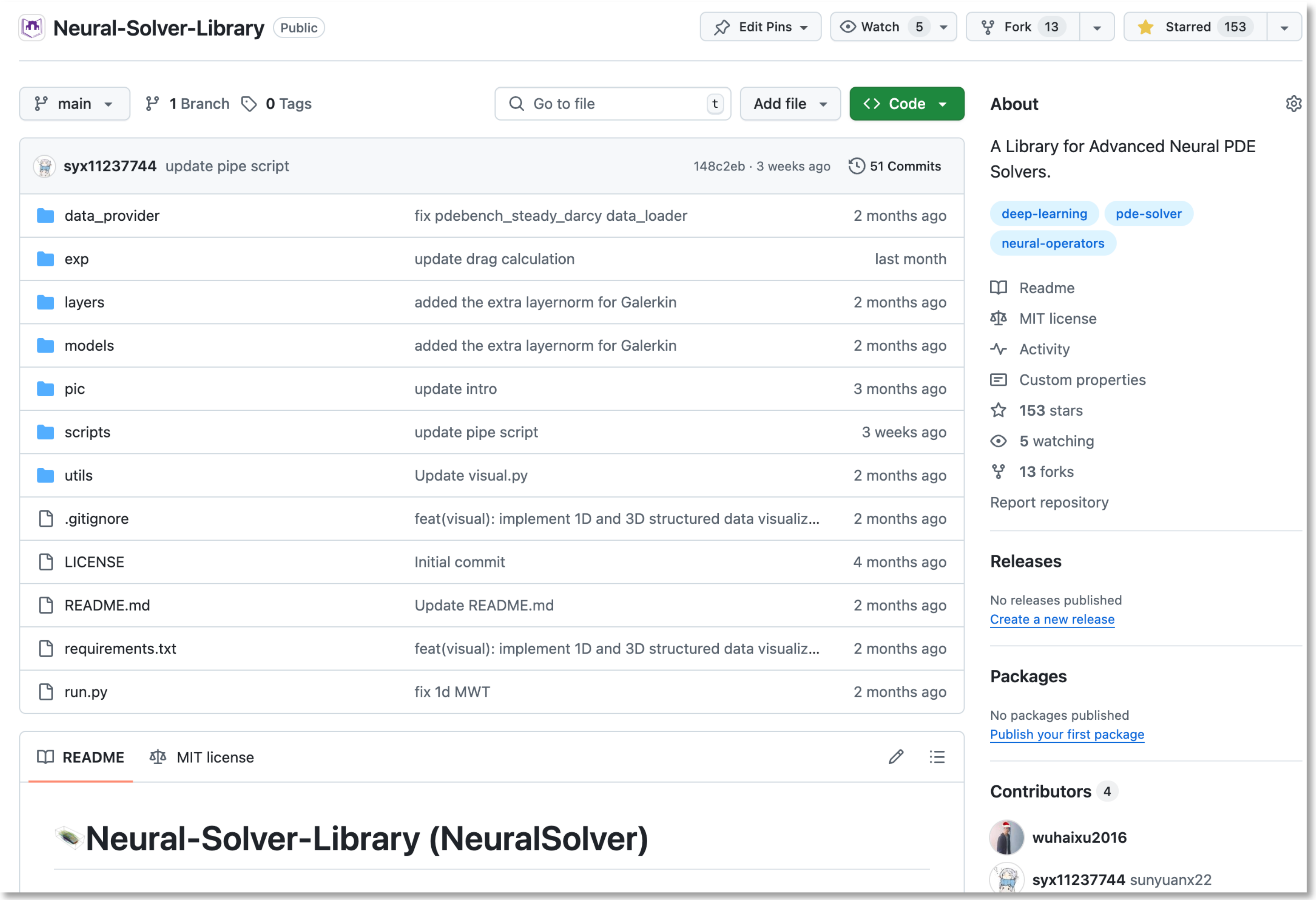


(a) Scaling of Input Resolution



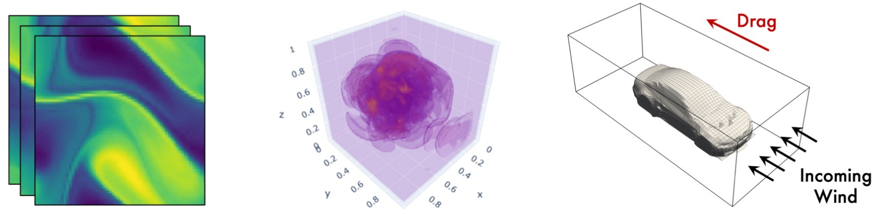
(b) Scaling of Evaluation Resolution

Neural-Solver-Library



Code Link: https://github.com/thuml/Neural-Solver-Library

- ✓ 17 different PDE solvers
- ✓ 6 standard benchmarks, PDEBench and design tasks



Task 1: Standard   Task 2: PDEBench   Task 3: ShapeNet Car

Welcome to join us and add a new feature to this Library!

Acknowledgement

Mingsheng Long

Wojciech Matusik

Jianmin Wang

NVIDIA

physicsnemo

SIEMENS

Hang Zhou

Yuezhou Ma

Huakun Luo

Haonan ShangGuan

Yuanxu Sun

Huikun Weng

From Transolver to Transolver-3:

Scaling Neural Solvers to Industrial-Scale Geometries

Haixu Wu

Computational Design and Fabrication Group, MIT CSAIL

Feb 04, 2026