From the Transolver Family to GeoPT:

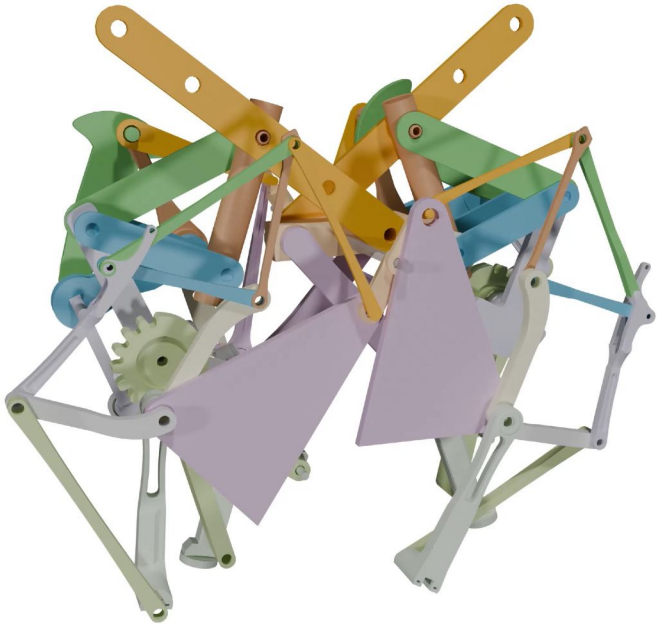
A Scaling Path for Neural Simulators

Haixu Wu

MIT CSAIL

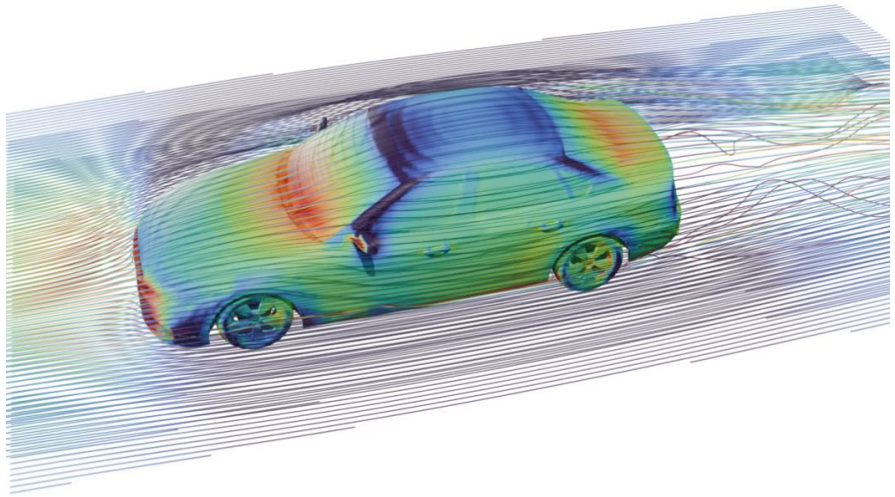
May 12, 2026

Physics Simulation for Scientific Engineering



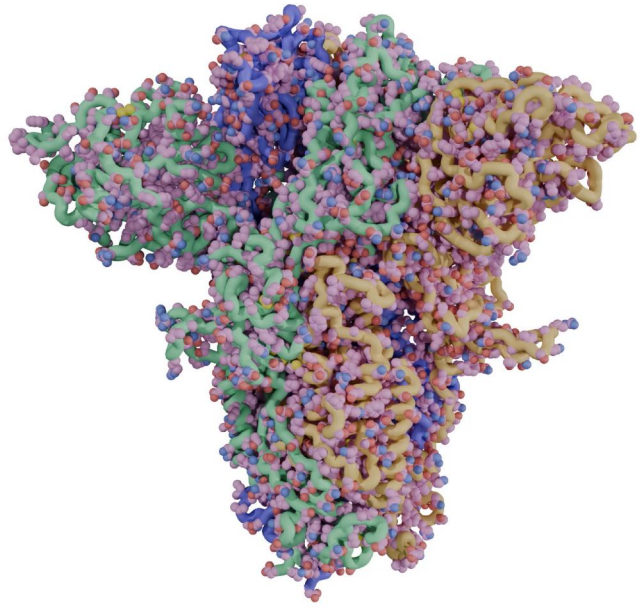
Rigid Body Dynamics

[Jansen's linkage]



Fluid Dynamics

[DrivaerML, 2024]



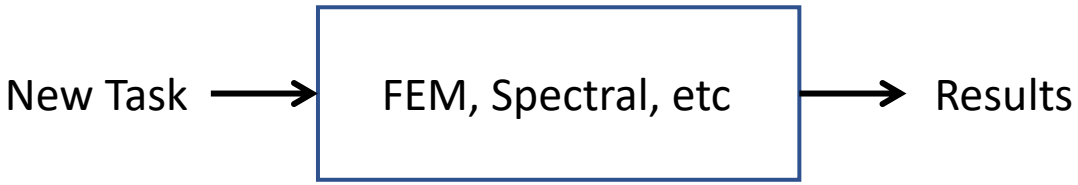
Protein Dynamics

[SARS-CoV-2]

Simulation is a foundation of industrial design and scientific discovery.

Physics Simulators

Classic Numerical Methods



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

Stable vs. Slow and Discretized

Ansys

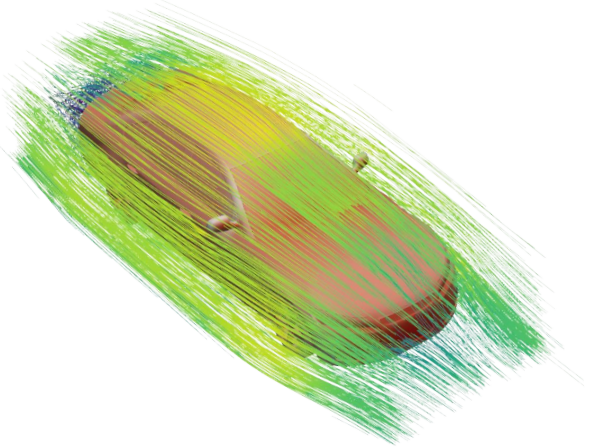
DASSAULT SYSTEMES

SIEMENS

HEXAGON

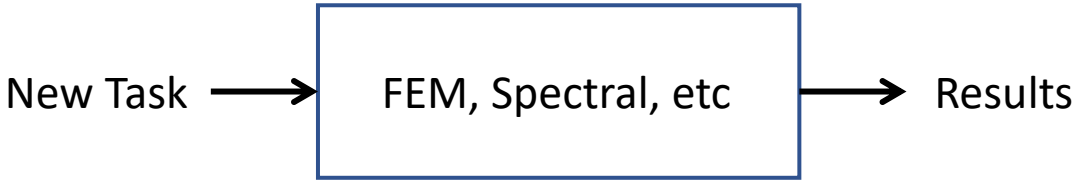
Discretized Mesh

6.1×10^4 CPU-hours



Physics Simulators

Classic Numerical Methods

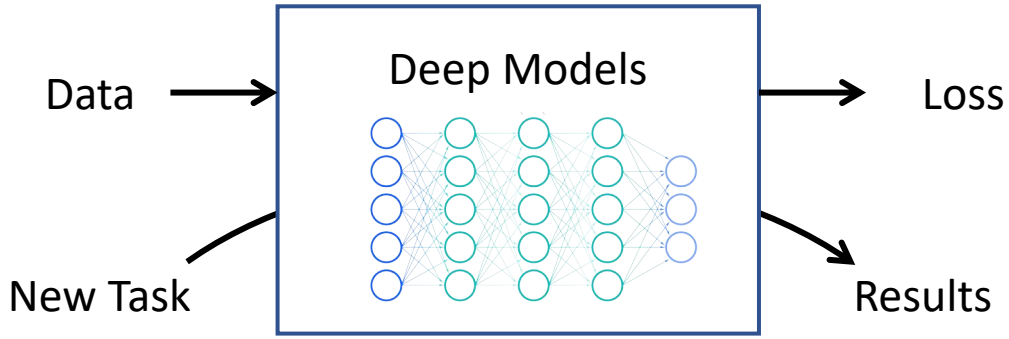


- Recalculation for every new sample
- Each round will incur huge costs

Stable vs. Slow and Discretized

Ansys DASSAULT SYSTEMES SIEMENS HEXAGON

Neural Simulators

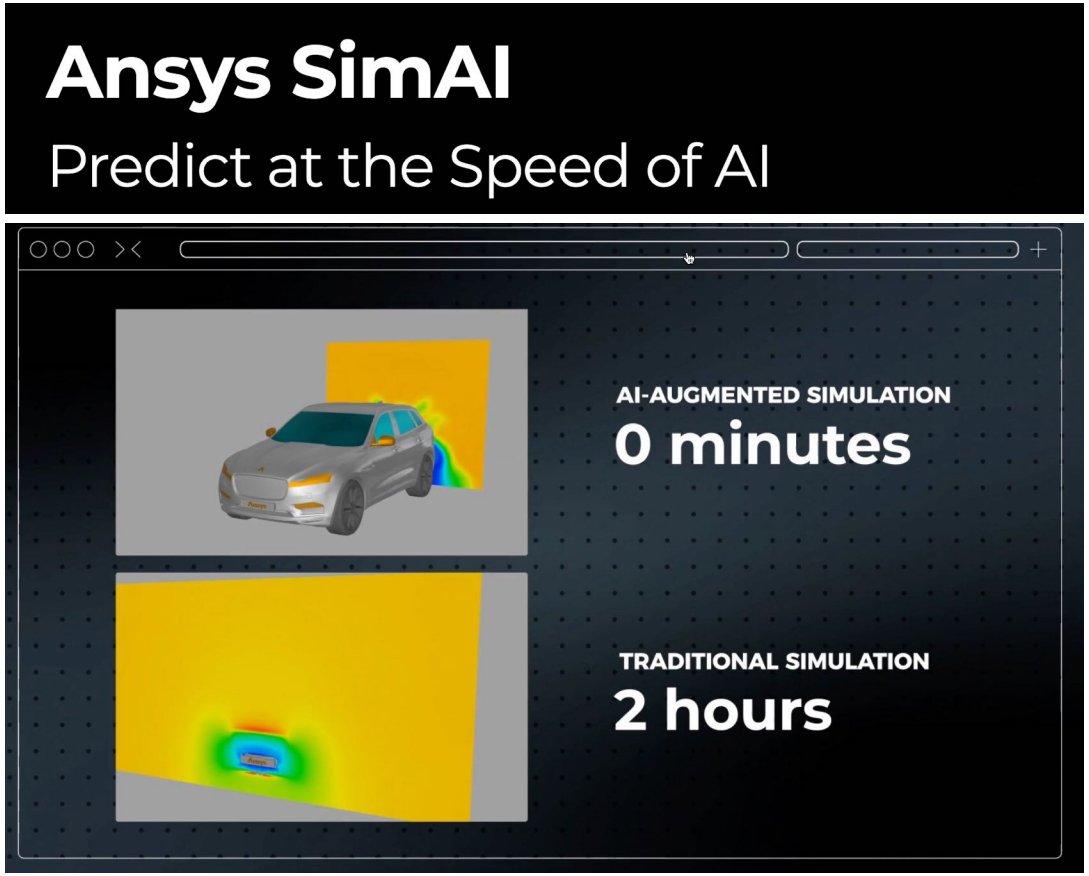


- Training once, inference a lot
- Each round needs several seconds

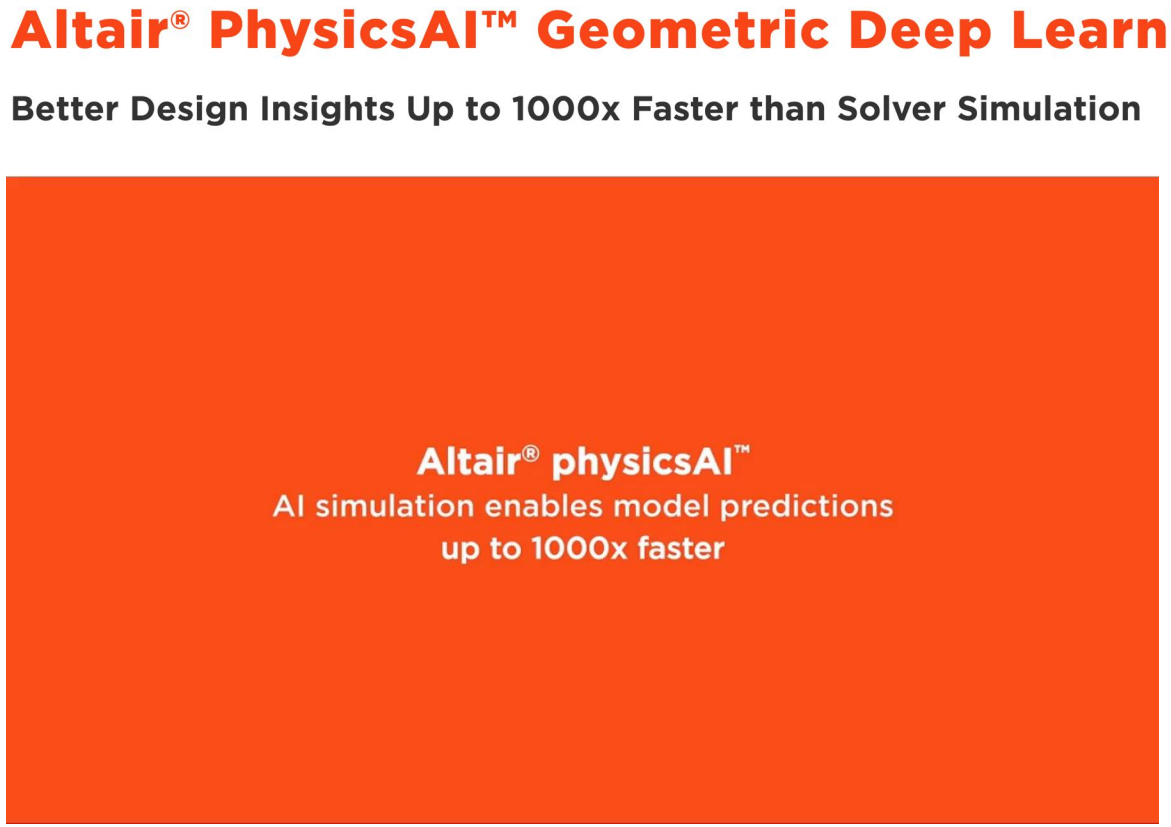
An efficient / precise surrogate tool

(Ideally)

A Valuable Direction

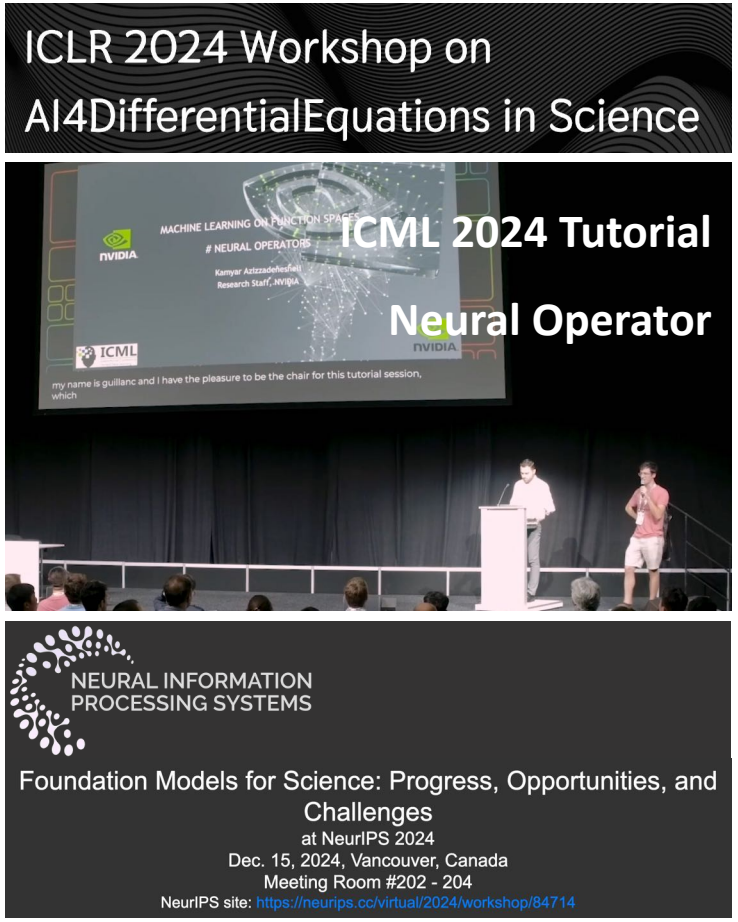


https://www.ansys.com/products/simai



https://altair.com/physicsai

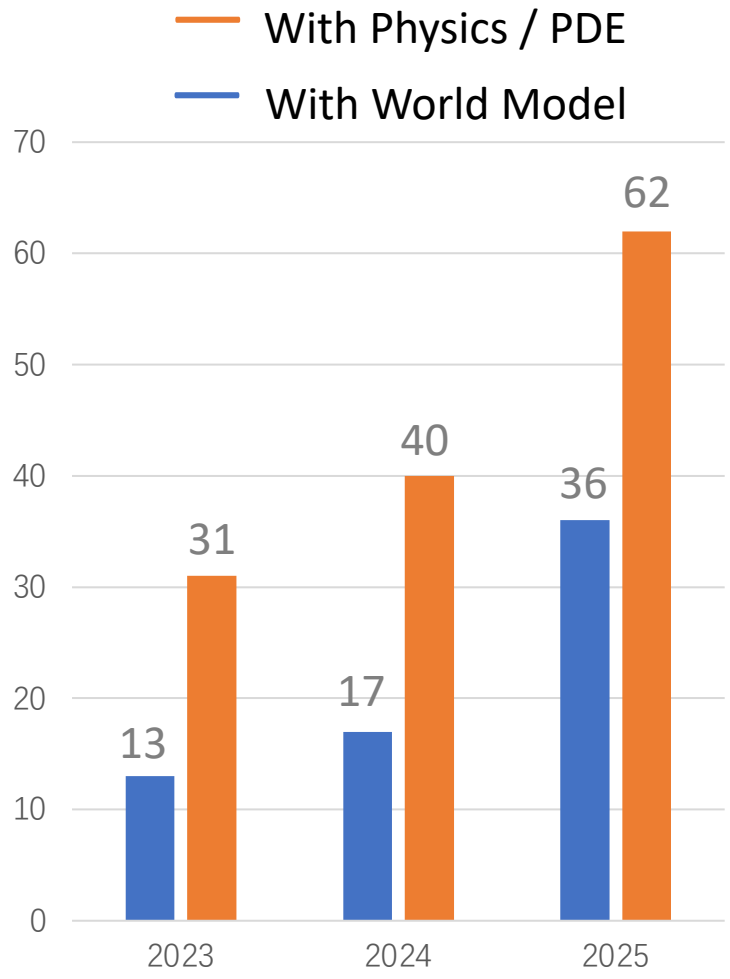
A Booming Direction



2024

ICLR 2025

2025



Accepted NeurIPS Papers

An Unsolved Direction

Realistic World

https://deepmind.google/models/genie/

Physical World

https://x.com/afshawnl

Why we need such paradigm shift

Efficiency? Numerical methods can be very good at Million-scale problems.

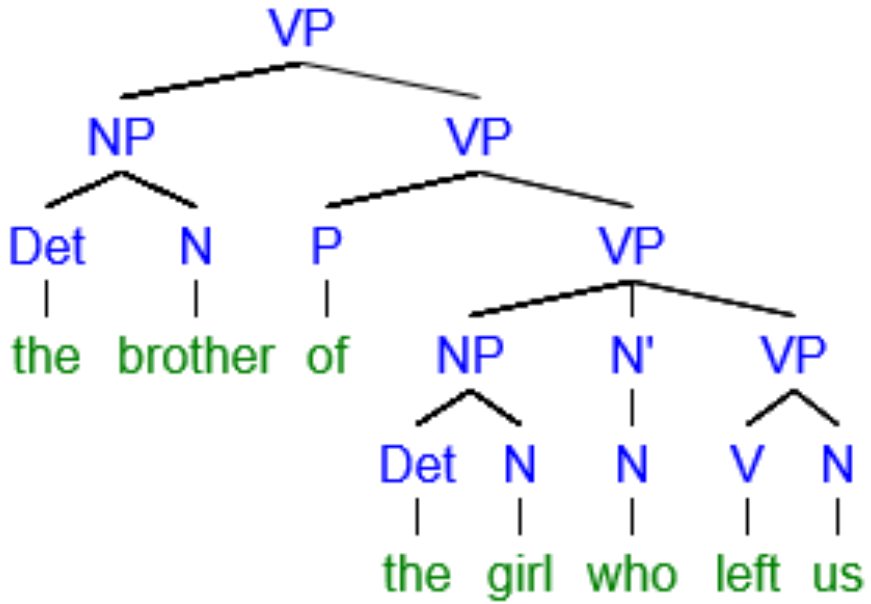
16M mesh points: 40.7 ms / step

4M mesh points: 11.1 ms / step

Sun et al., Leapfrog Flow Maps for Real-Time Fluid Simulation, SIGGraph 2025

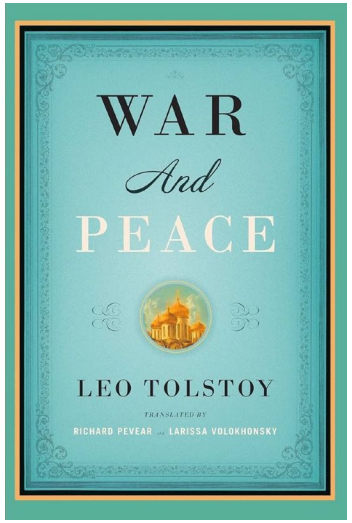
Why we need such paradigm shift

Recap LLM community: semantic recognition.



NLP before deep learning

(Structured and principled)

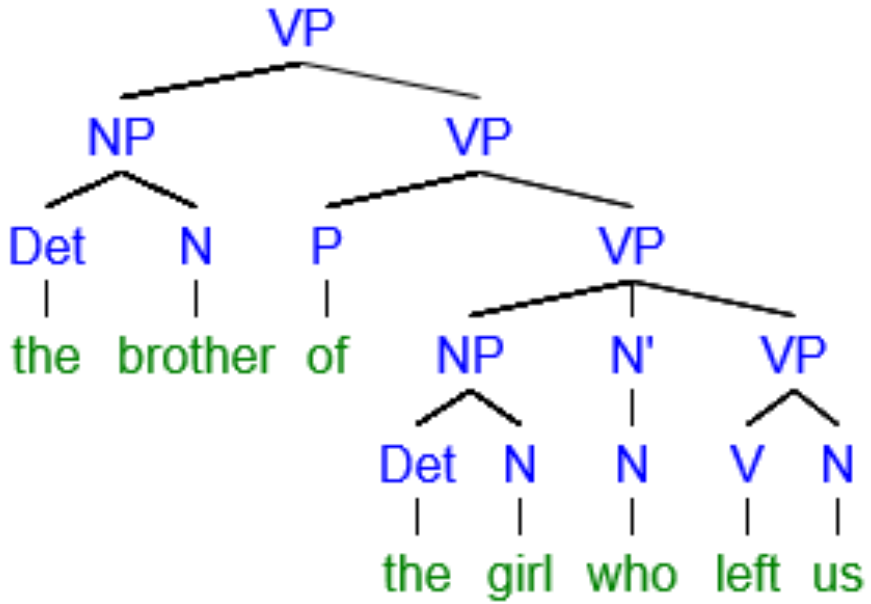


$\approx$ 0.6M words

What will happen?

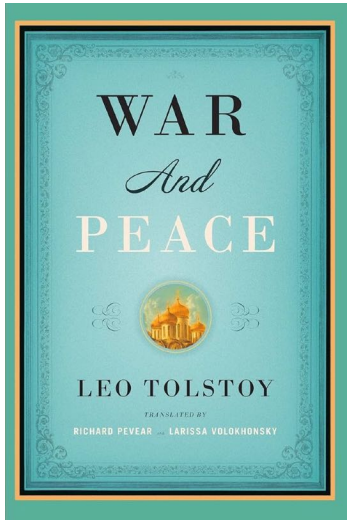
Why we need such paradigm shift

Recap LLM community: semantic recognition.



NLP before deep learning

(Structured and principled)



≈ 0.6M words

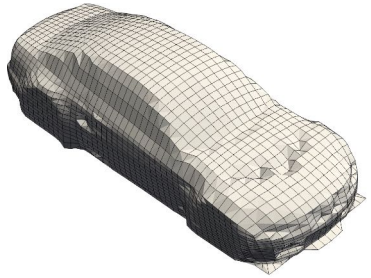
What will happen?

(1) Sharply increased complexity

(2) Rich semantic ~~(life)~~ knowledge

Why we need such paradigm shift

Neural simulators' advantage emerges when the problem complexity scales beyond the effective scope of numerical methods.

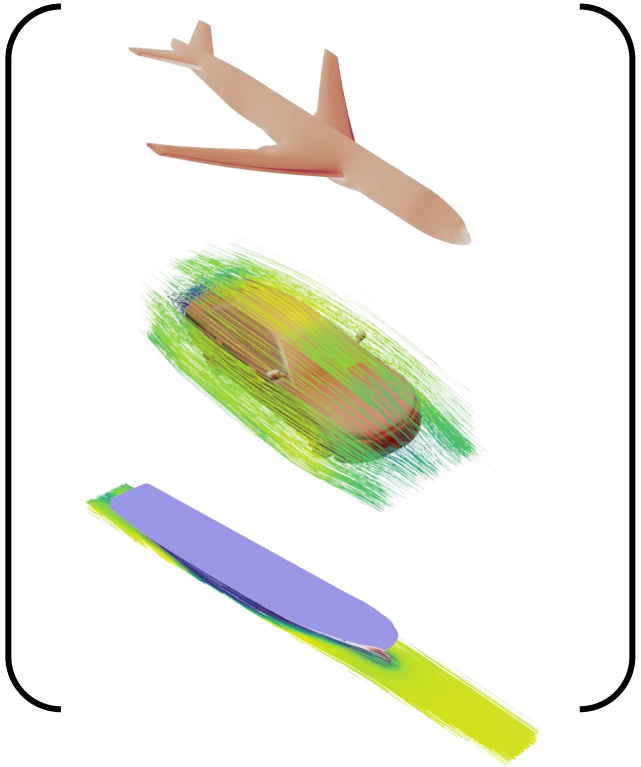


Single task

Small mesh

Scaling Task

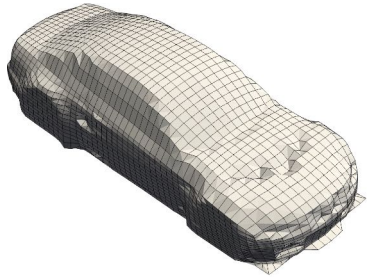
Scaling Geometry



Multi-task / Large mesh

(100-million-scale)

Scaling Path for Neural Simulators



Single task

Small mesh

Unisolver
(ICML 2025)

GeoPT
(ICML 2026)

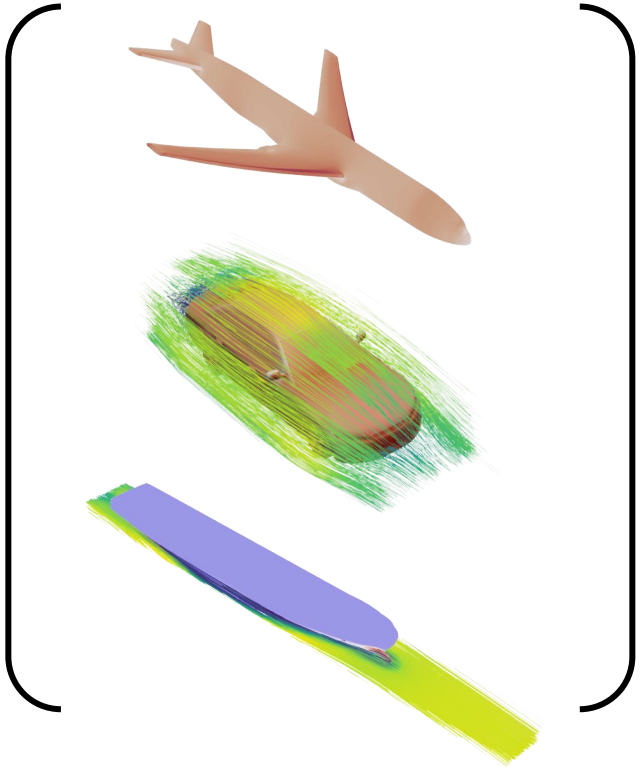
Scaling Task

Scaling Geometry

Transolver
(ICML 2024)

Transolver++
(ICML 2025)

Transolver-3
(ICML 2026)



Multi-task / Large mesh

(100-million-scale)

Scaling Path for Neural Simulators

Scalable Backbone – – – – – – **General Geometry** — Transolver

Geometry Scaling – – – – – – – Industrial-scale Mesh — Transolver++, Trasolver-3

Physics Scaling – – – – – – – – – Large-scale Pre-training — GeoPT

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

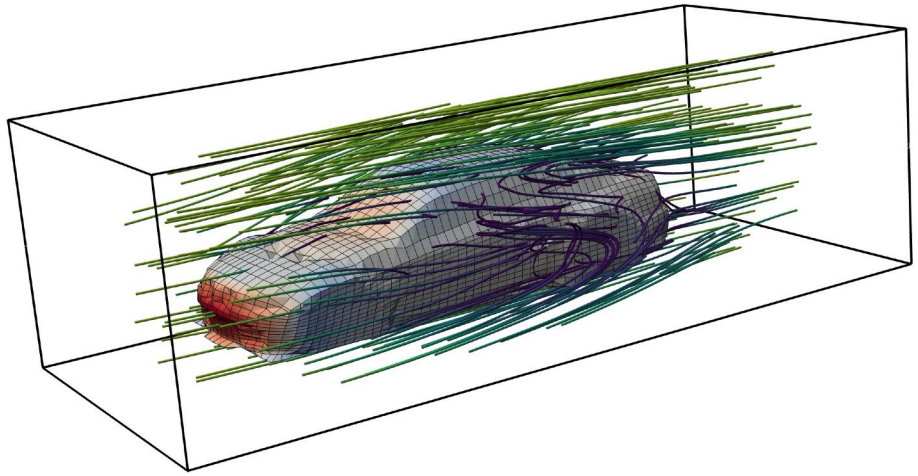
Haowen Wang

Jianmin Wang

Mingsheng Long

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

Challenges in Industrial Physics Simulation

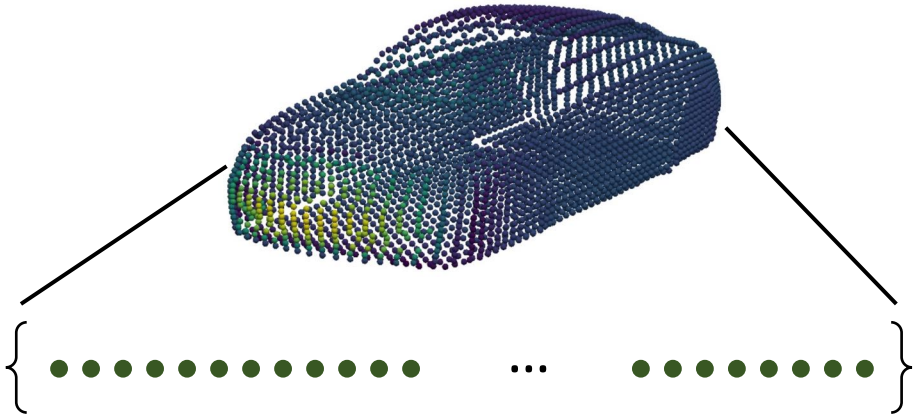


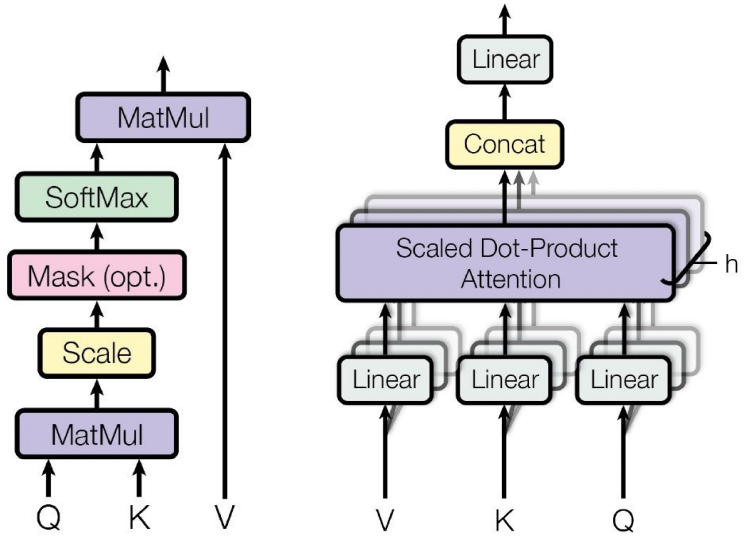
Task: Estimate the drag coefficient of a given shape:

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 Simulators

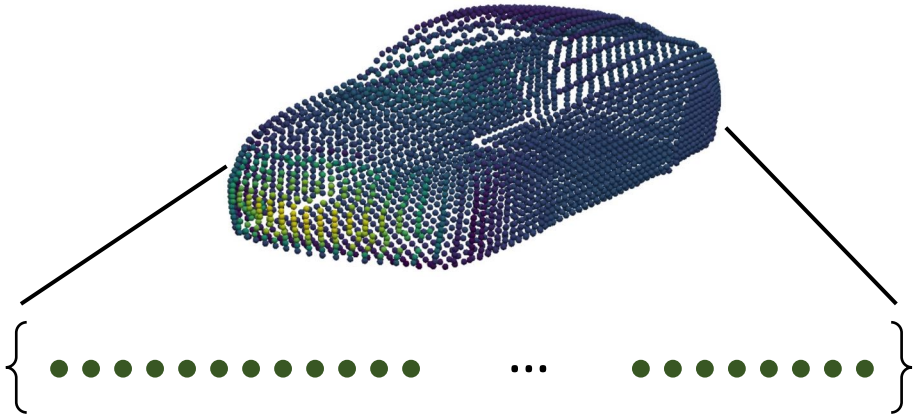


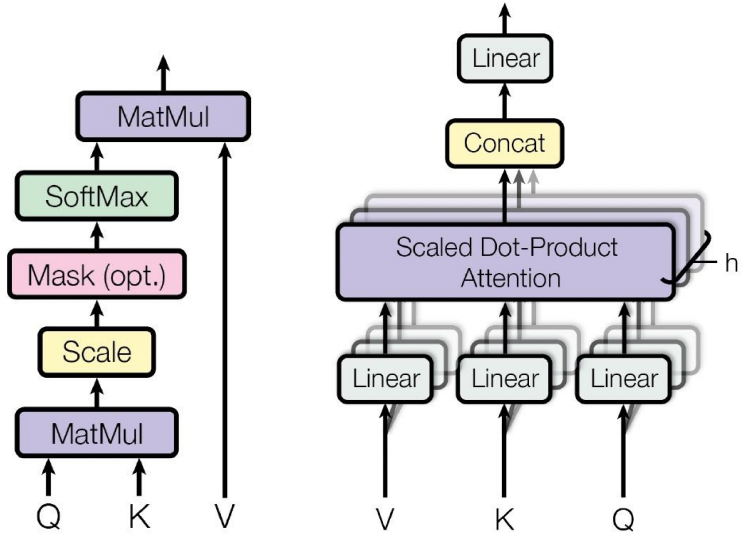


(1) Geometries as point sequences (2) Attention as Monte Carlo Integral

OFormer, Galerkin Transformer, GNOT, etc

Transformer-based Simulators



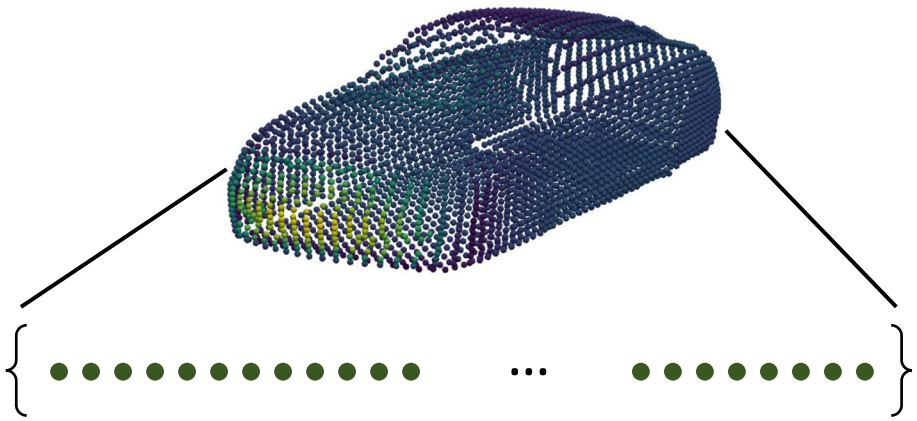


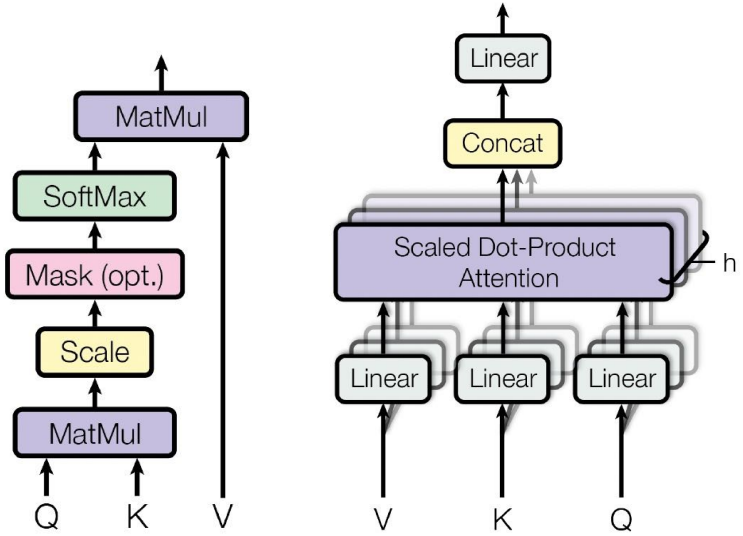
(1) Geometries as point sequences (2) Attention as Monte Carlo Integral

OFormer, Galerkin Transformer, GNOT, etc

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

Transformer-based Simulators





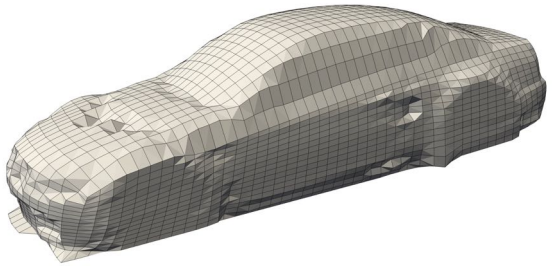
(1) Geometries as point sequences (2) Attention as Monte Carlo Integral

OFormer, Galerkin Transformer, GNOT, etc

How to efficiently capture physical correlations underlying discretized meshes

is the key to "transform" Transformers into practical neural simulators

A Foundational Idea of Transolver



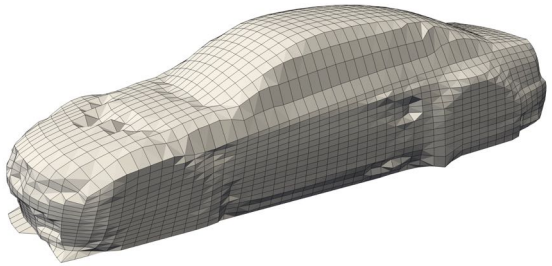
Discretized Domain

Previous Work

Being “trapped” to superficial and unwieldy meshes

Difficulties in Complexity, Geometry, Physics

A Foundational Idea of Transolver

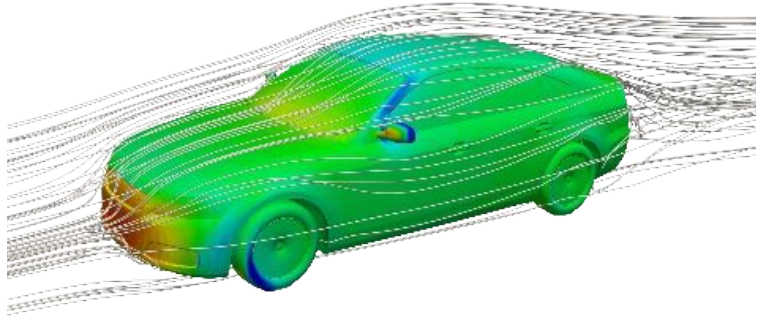


Discretized Domain

Previous Work

Being “trapped” to superficial and unwieldy meshes

Difficulties in Complexity, Geometry, Physics



Physics Domain

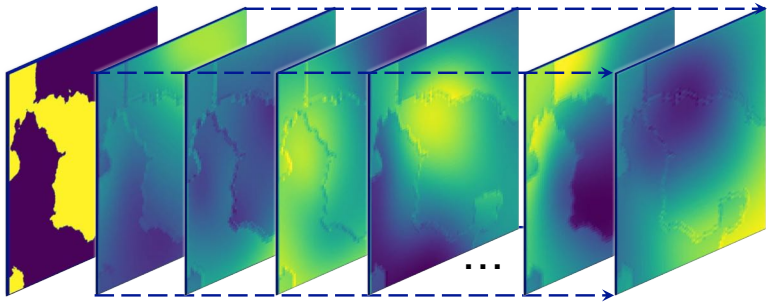
Transolver

Learning **intrinsic physical states** underlying

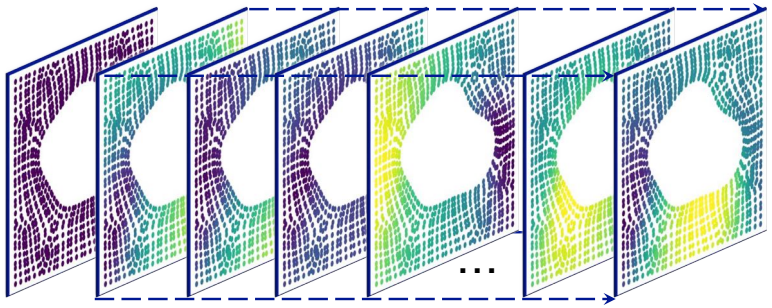
complex and large-scale geometries

Better Efficiency, Geometry, Physics Modeling

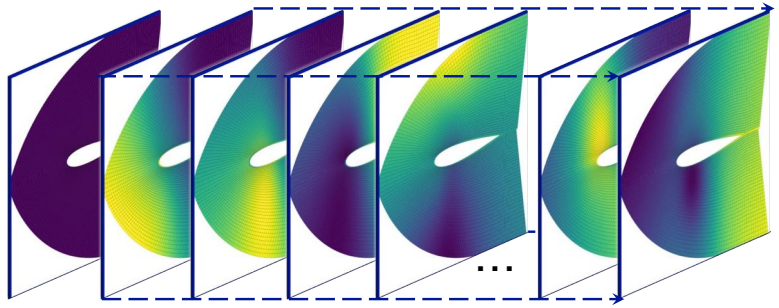
Learning Physical States



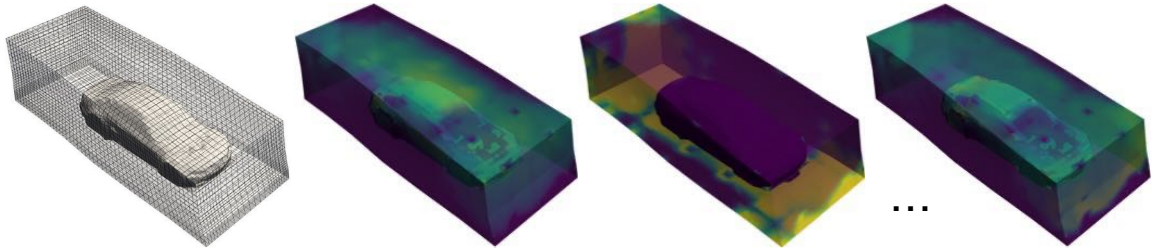
(a) Slices for Darcy, 2D Regular Grid



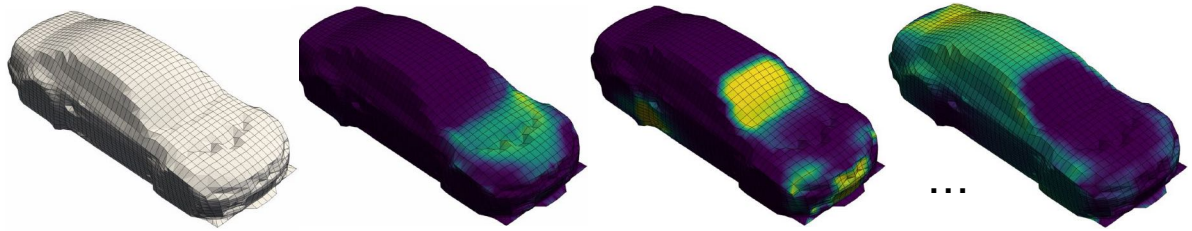
(b) Slices for Elasticity, 2D Point Cloud



(c) Slices for Airfoil, 2D Mesh



(d) Slices for Shape-Net Car Surrounding Velocity, 3D Volumes

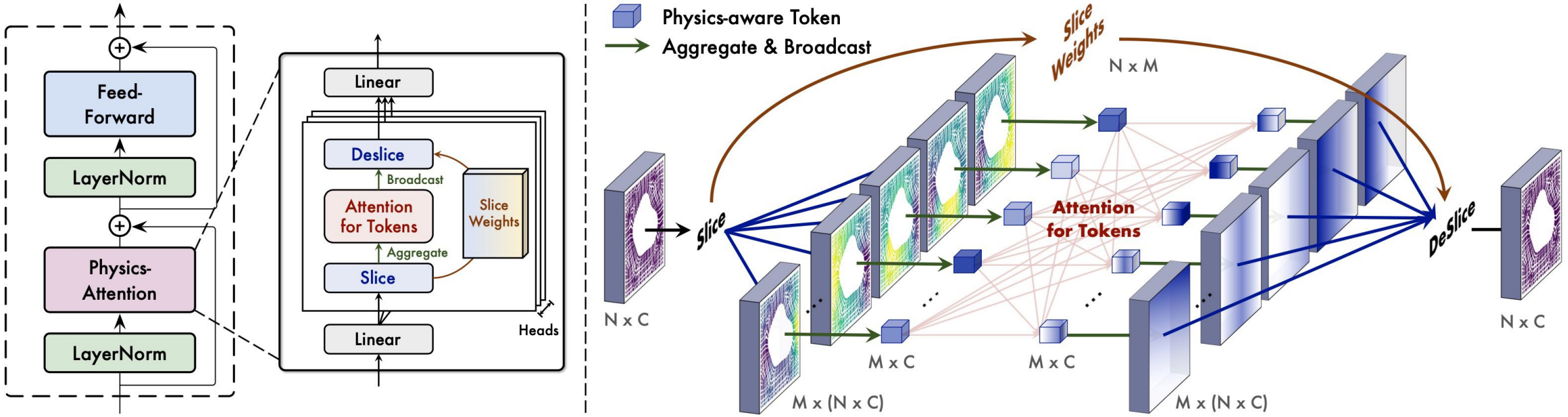


(e) Slices for Shape-Net Car Surface Pressure, 3D Mesh

Mesh points under **similar physical states** will be ascribed to the same **slice**

and then encoded into a **physics-aware token**.

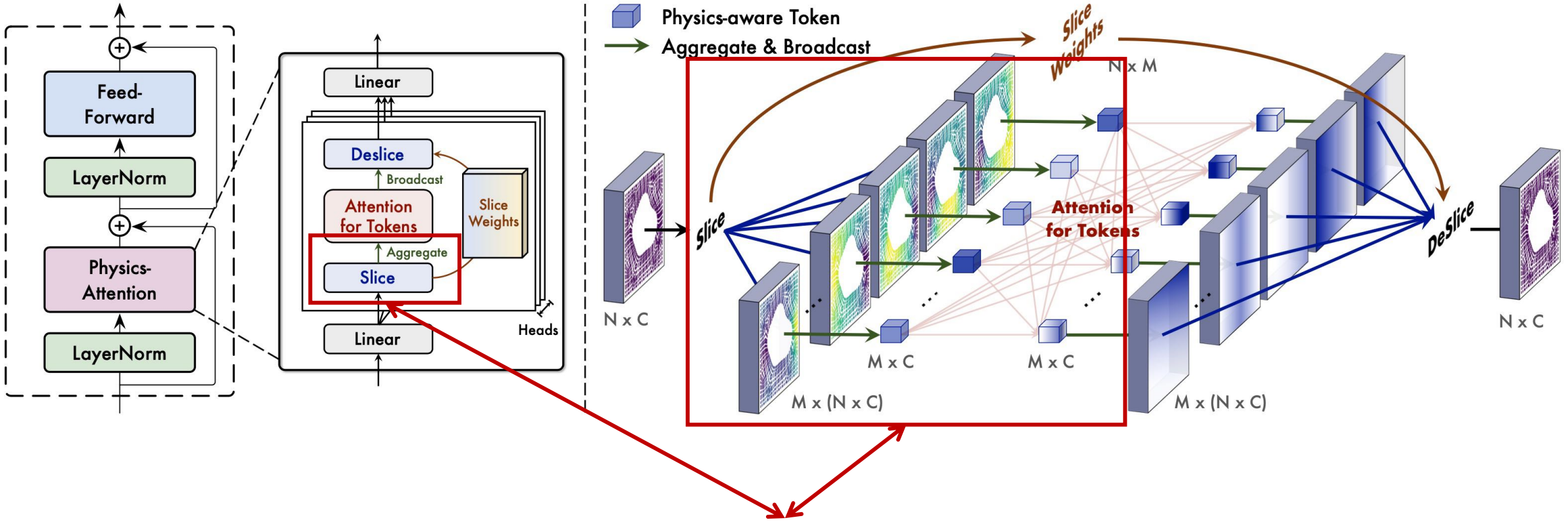
Overview of Transolver



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

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

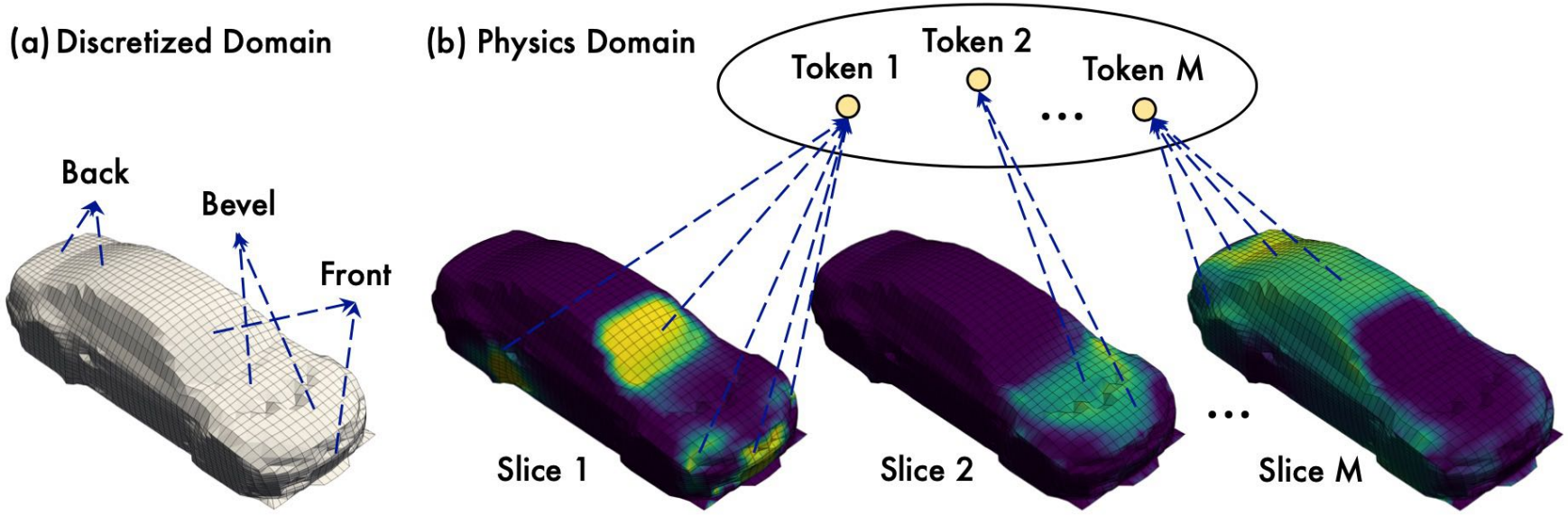
Step 1: Mesh → Physics



① Mesh → physics

To obtain physics-aware tokens

Learning Physics-aware Tokens



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

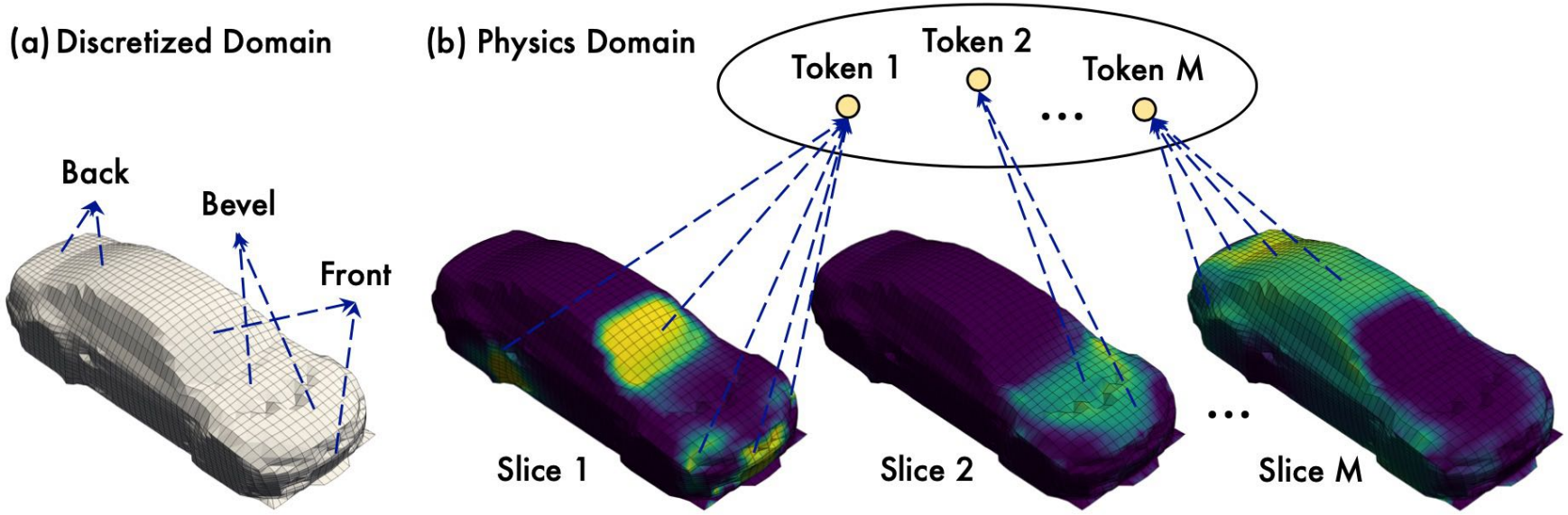
$$\{\mathbf{w}_i\}_{i=1}^{N} = \{\underline{\text{Softmax}}\left(\text{Project}\left(\mathbf{x}_i\right)\right)\}_{i=1}^{N}$$

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

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

Softmax for low-entropy slices

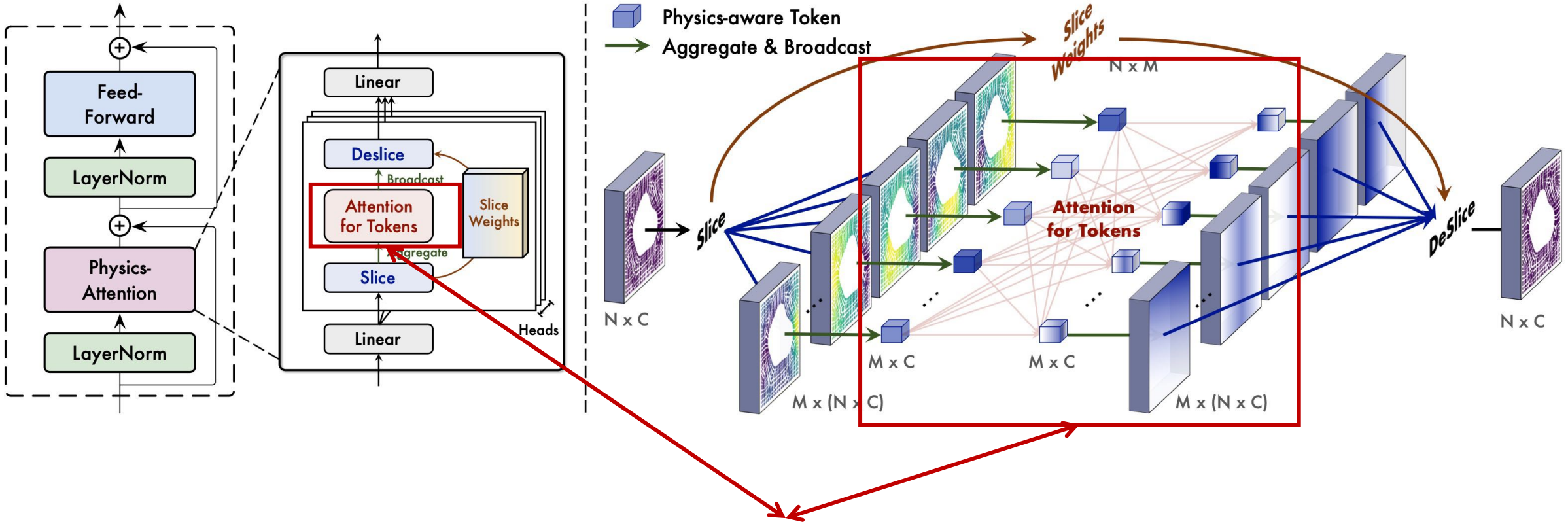
Learning Physics-aware Tokens



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

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

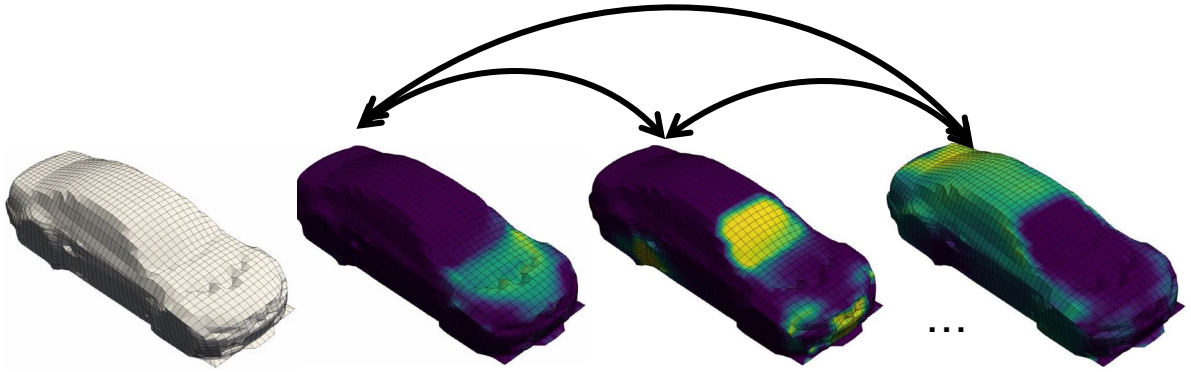
Step 2: Physics Interaction



② Attention among physics tokens

Approximate Integral to enable simulations

Attention among physics tokens

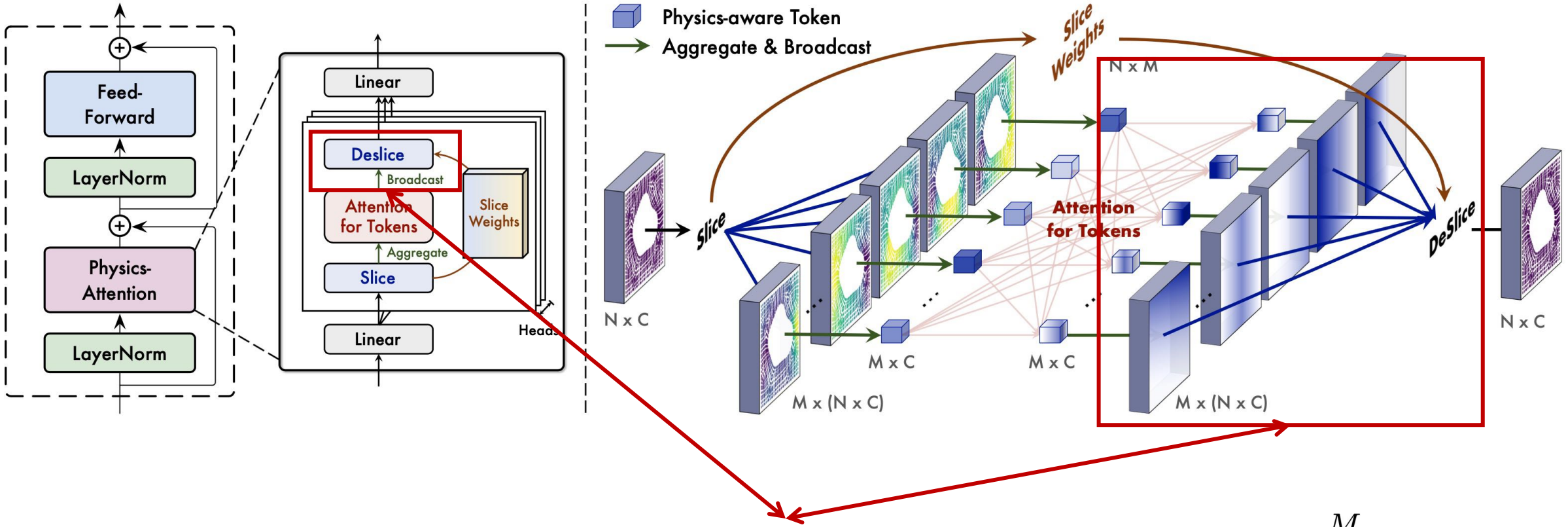


$$\mathbf{q}, \mathbf{k}, \mathbf{v} = \mathrm{Linear}(\mathbf{z}), \quad \mathbf{z}' = \mathrm{Softmax}\left(\frac{\mathbf{q}\mathbf{k}^{\mathsf{T}}}{\sqrt{C}}\right)\mathbf{v}$$

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.

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

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

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

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

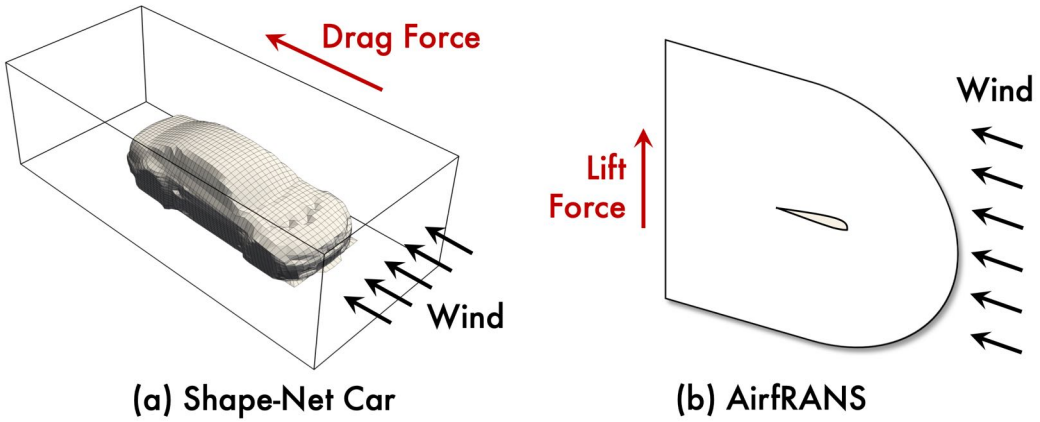
All the designs can be directly derived.

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

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

Experiments

Geometry	Benchmarks	#Dim	#Mesh
Point Cloud	Elasticity	2D	972
Structured Mesh	Plasticity	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



(a) Shape-Net Car (b) AirfRANS

Six standard benchmarks, two practical design tasks

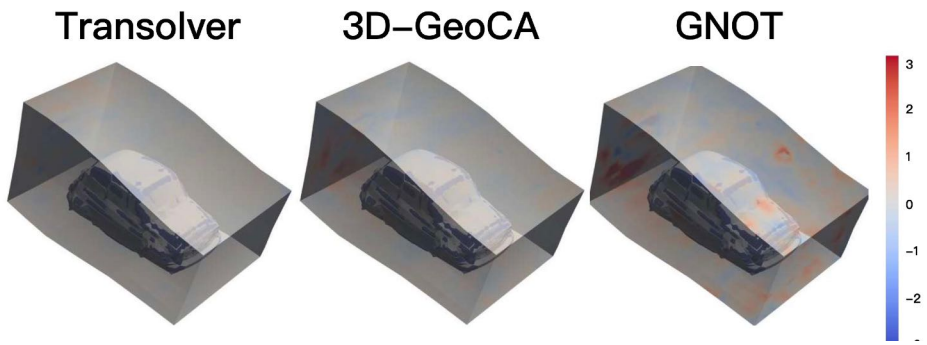
More than 20 baselines

Car and Airfoil Design

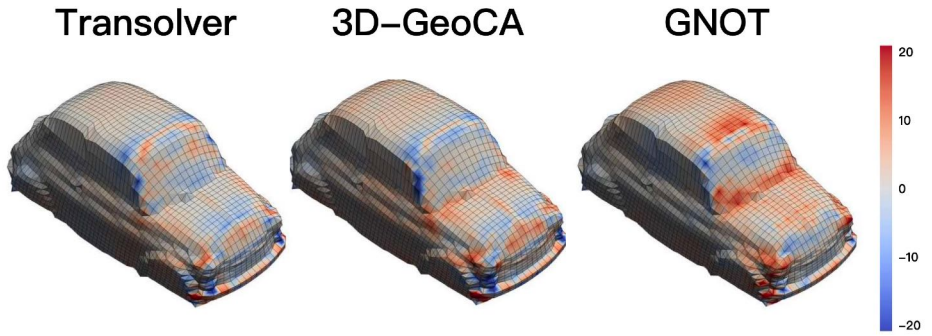
Models	Shape-Net Car				AirfRANS			
	Volume ↓	Surf ↓	C_D ↓	ρ_D ↑	Volume ↓	Surf ↓	C_L ↓	ρ_L ↑
Simple MLP	0.0512	0.1304	0.0307	0.9496	0.0081	0.0200	0.2108	0.9932
GraphSAGE [197]	0.0461	0.1050	0.0270	0.9695	0.0087	0.0184	0.1476	0.9964
PointNet [196]	0.0494	0.1104	0.0298	0.9583	0.0253	0.0996	0.1973	0.9919
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**

Model capability in "ranking" designs

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

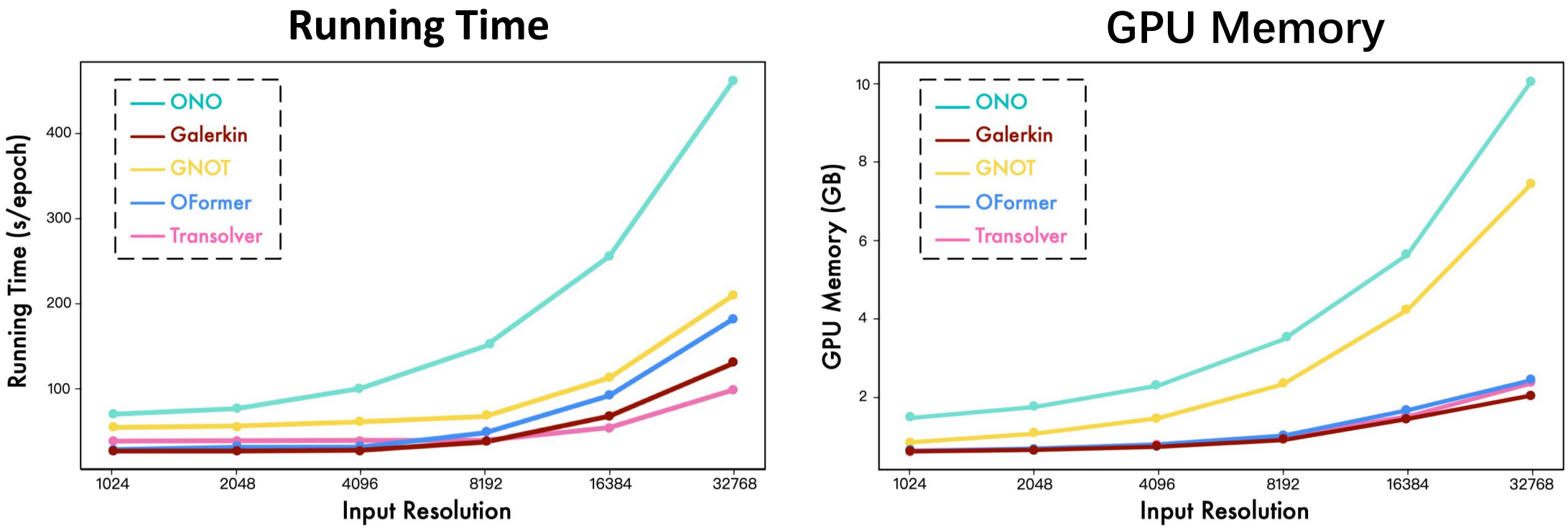


Surrounding Velocity Error Map



Surface Pressure Error Map

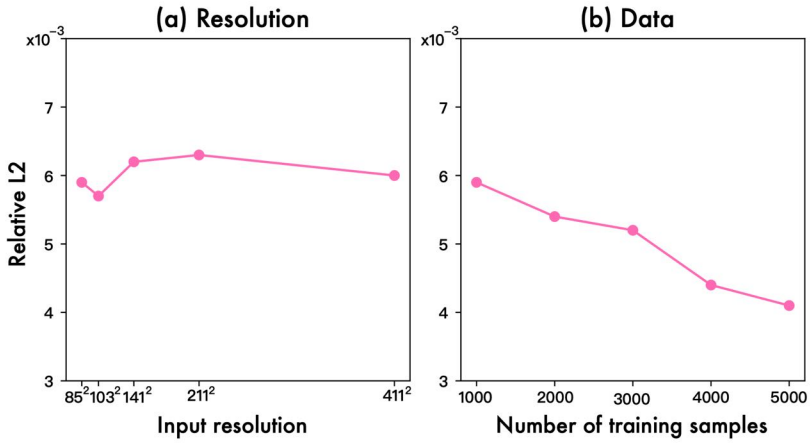
Efficiency



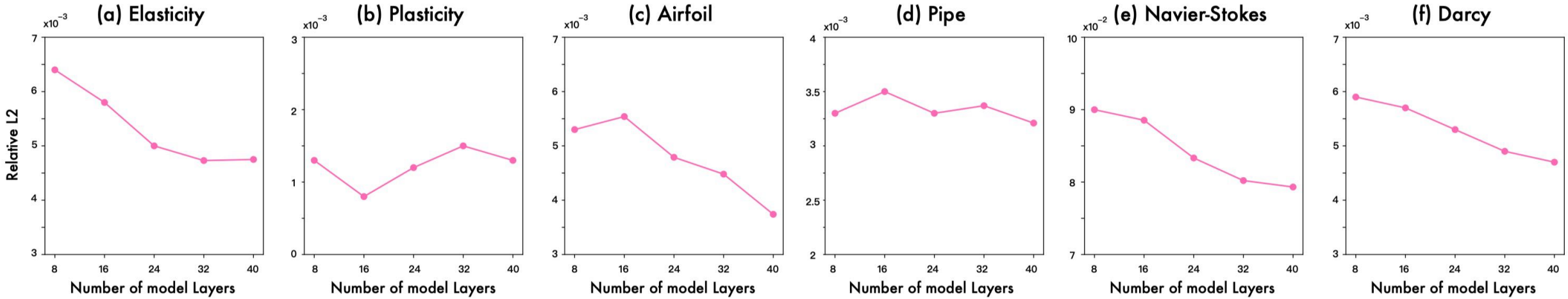
Favorable efficiency and performance balance

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

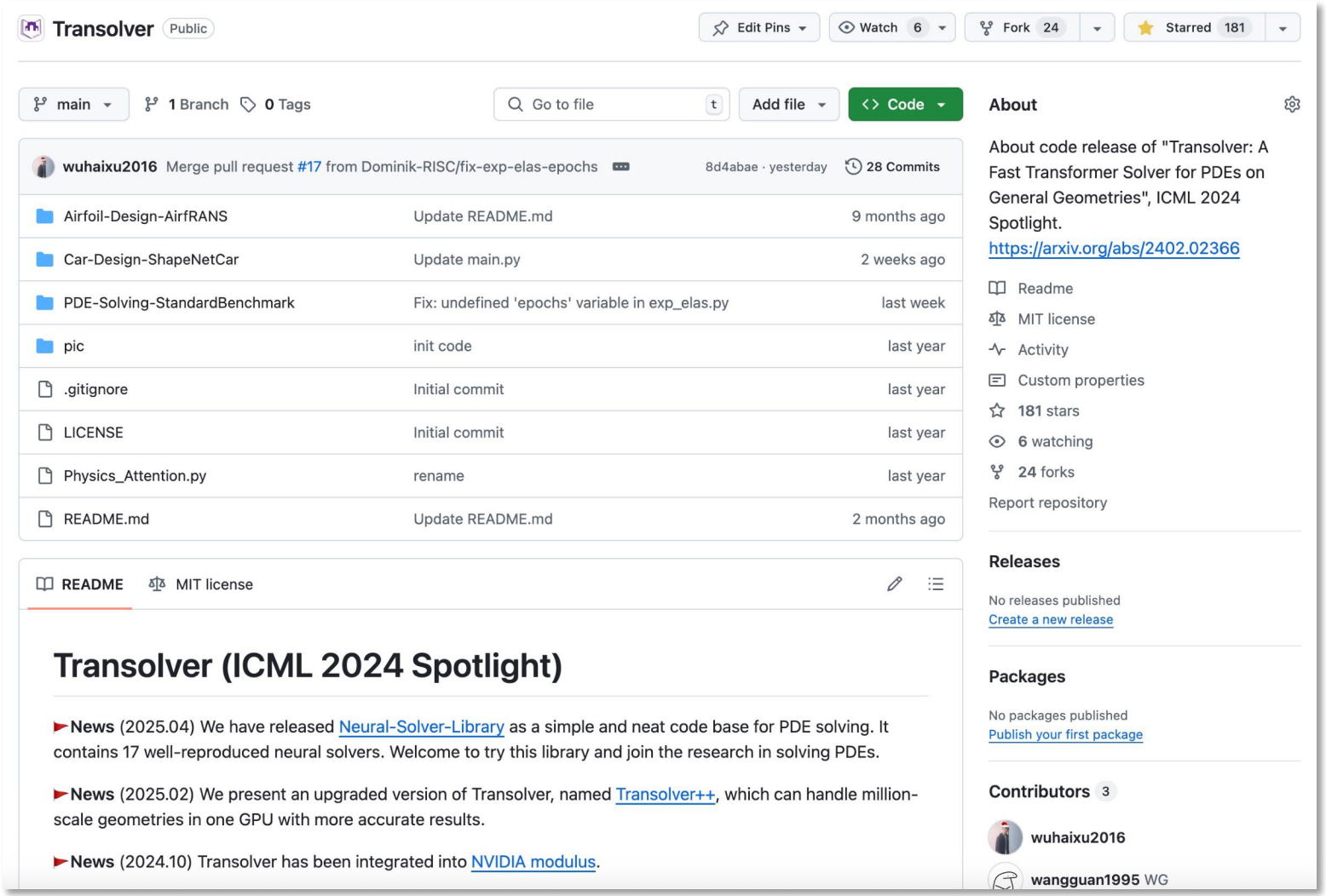
Pursuing PDE Foundation Models: Scalability



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



Open-Source Code



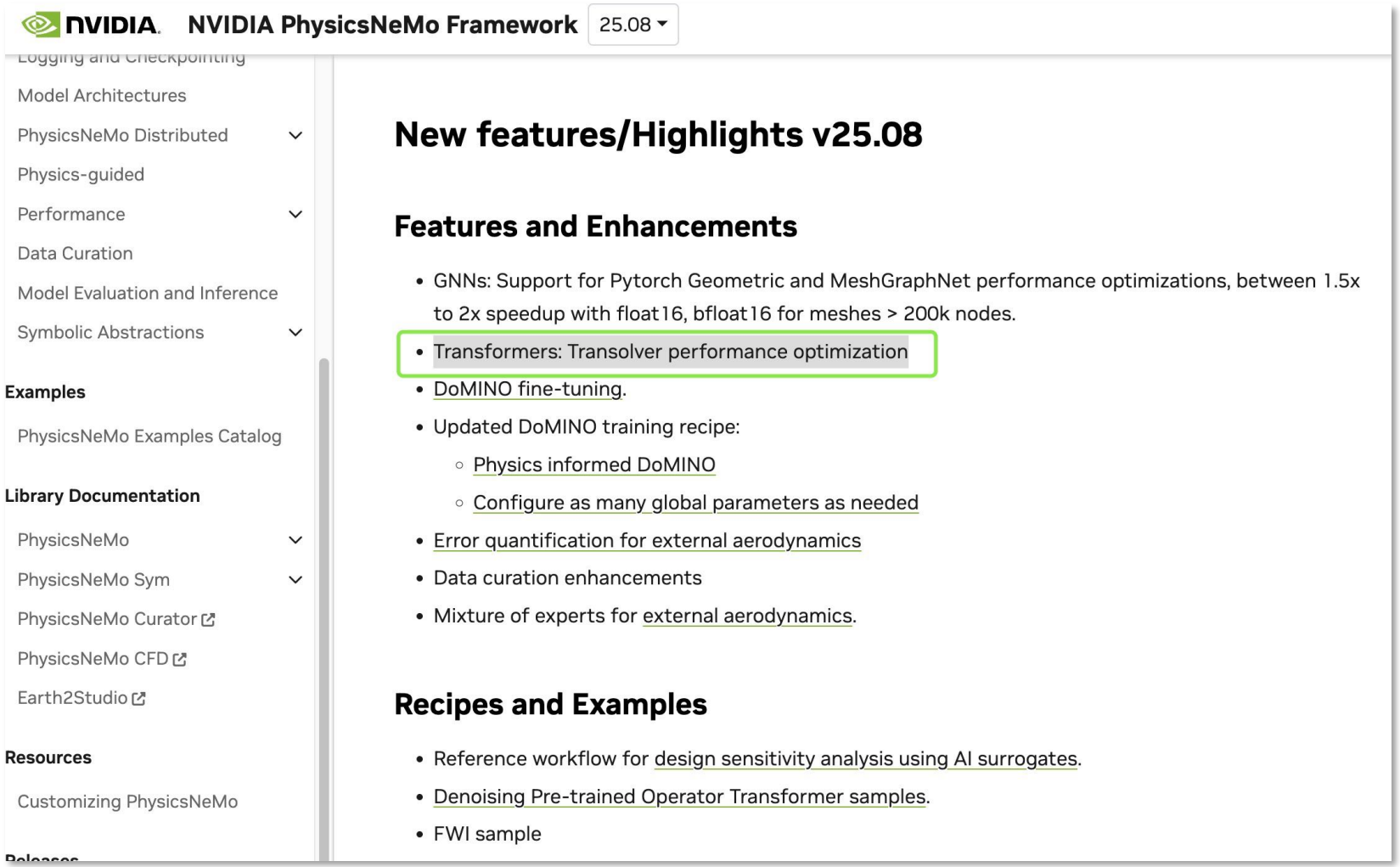
Code Link: *https://github.com/thuml/Transolver*

NVIDIA physicsnemo

Code for Transolver in Physicsnemo

Code for Transolver

NVIDIA PhysicsNeMo

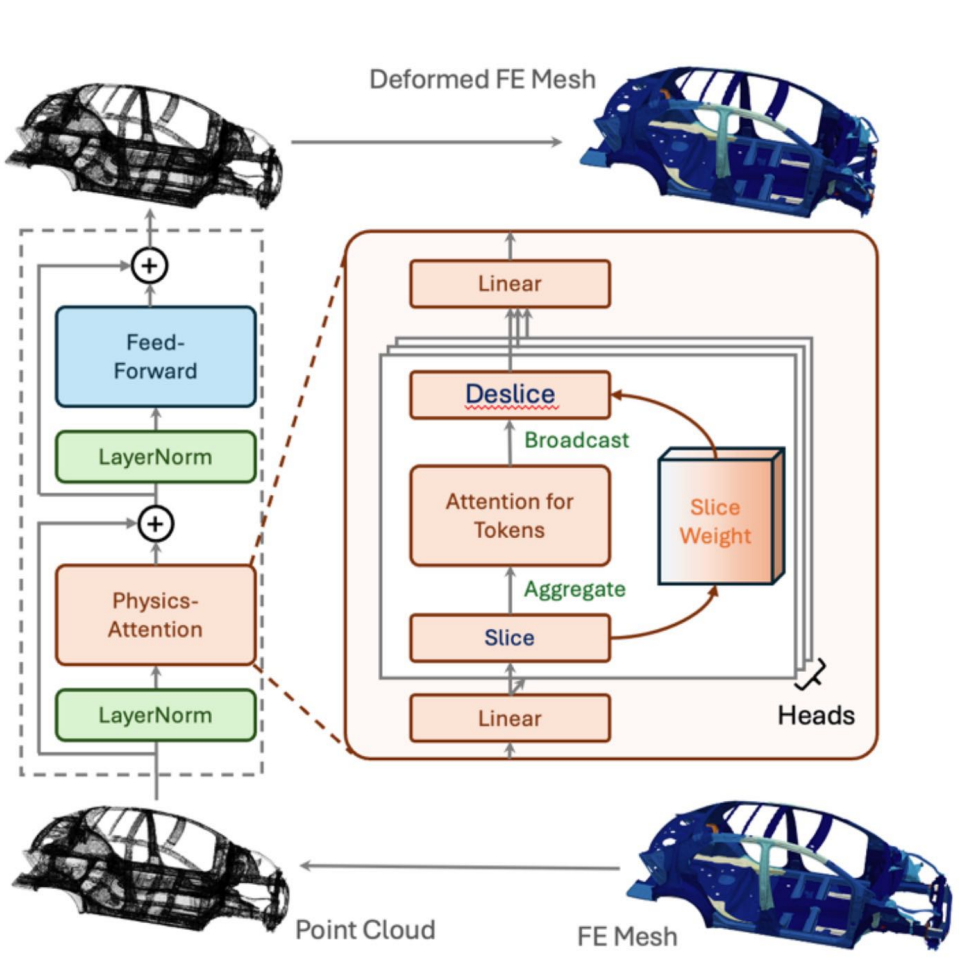


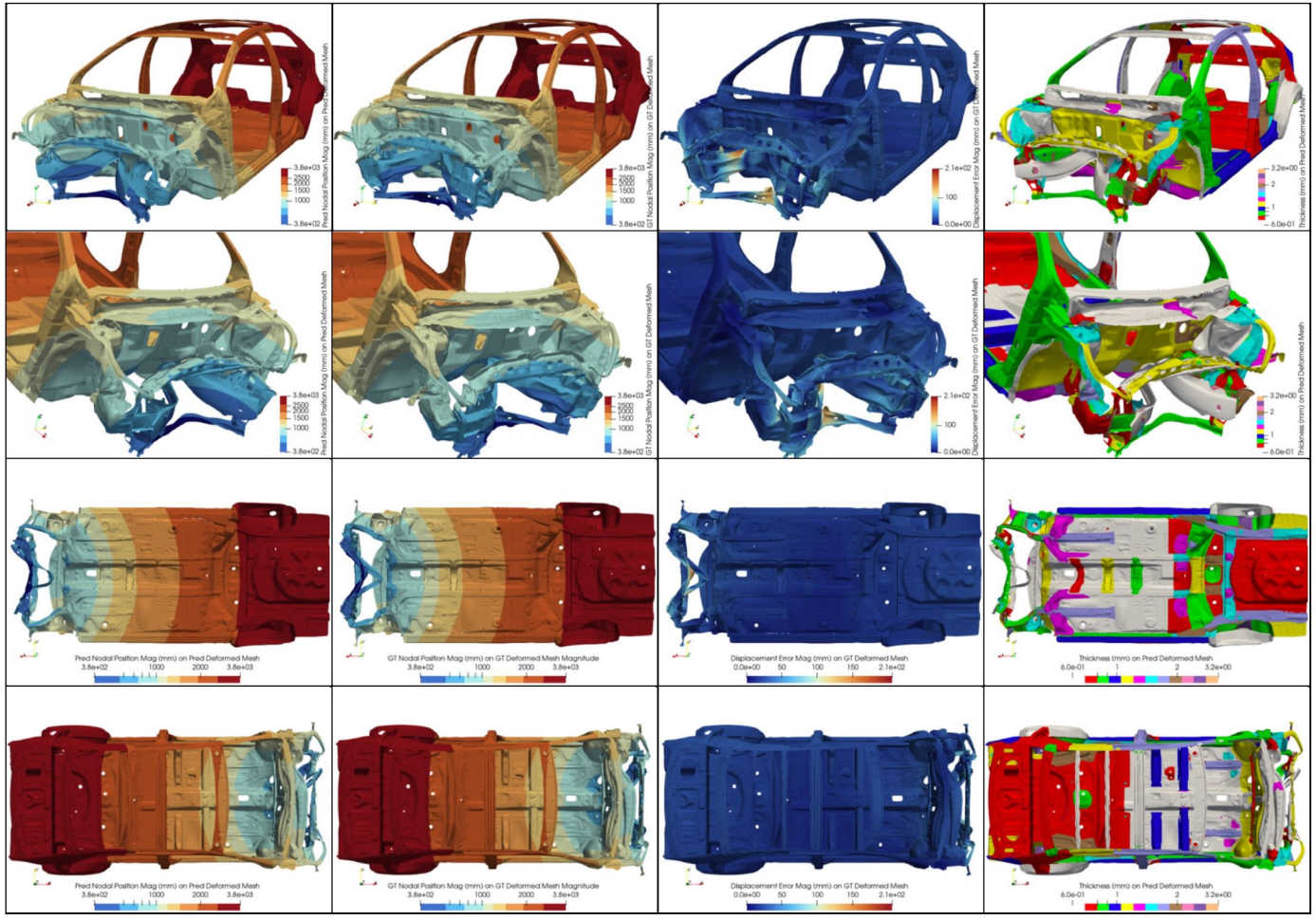
physicsnemo

"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

NVIDIA GeoTransolver

Luminary Adopts NVIDIA GeoTransolver AI Model Architecture, Unlocking Up to 10x Greater Accuracy in Physics AI

Luminary Cloud, Inc
March 16, 2026 • 4 min read

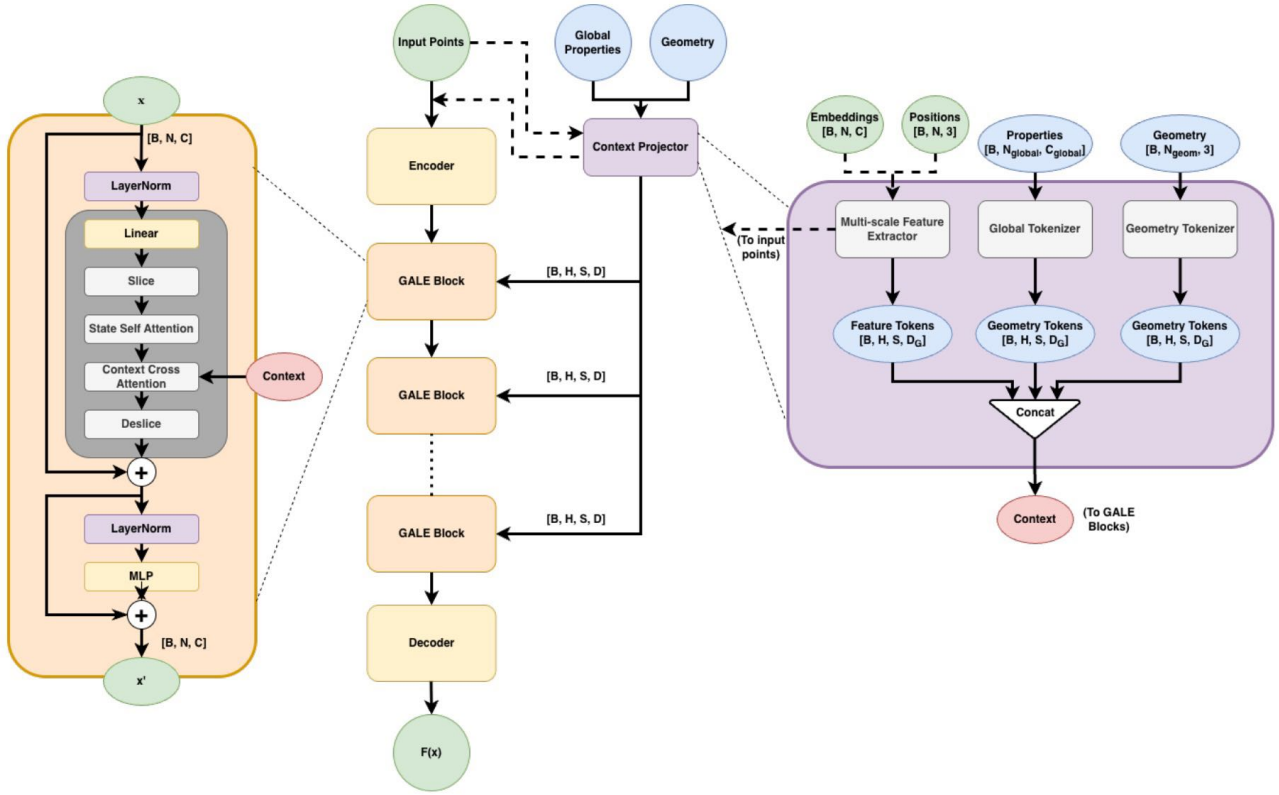
NVDA -1.03% ☆

Trade NVIDIA on Coinbase Trading disclosure ⓘ

luminary

Luminary Cloud, Inc

Near-simulation-level precision at AI scale supports advanced engineering programs, including Sceye's stratospheric platforms



Adams et al., GeoTransolver: Learning Physics on Irregular Domains Using Multi-scale Geometry Aware Physics Attention Transformer, arXiv 2025

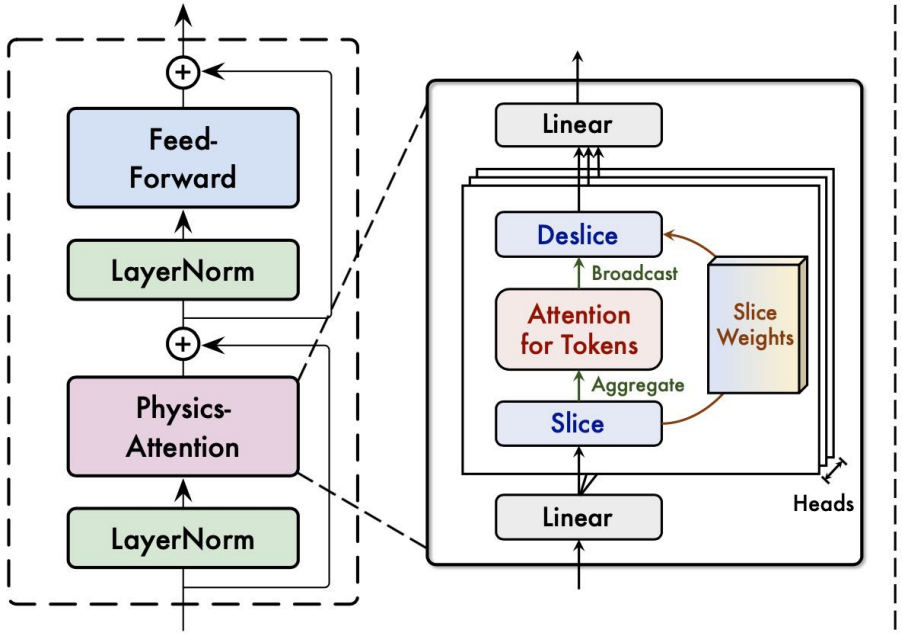
Scaling Path for Neural Simulators

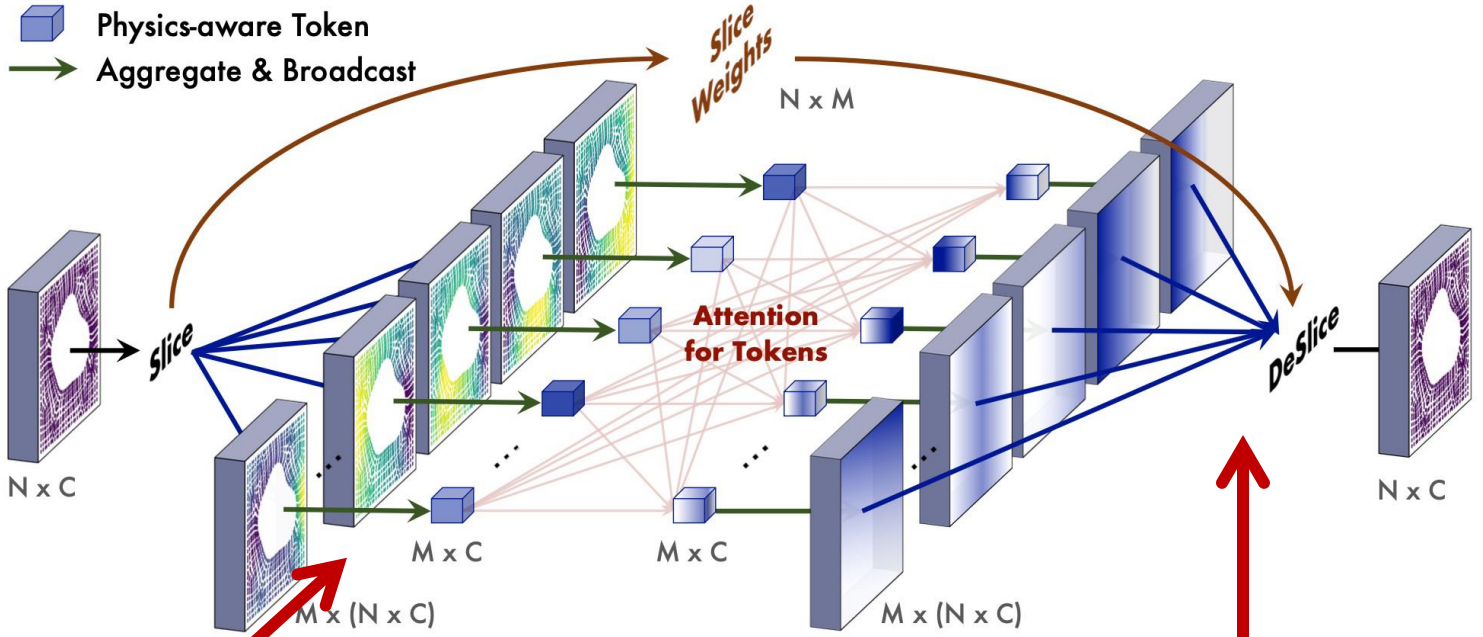
Scalable Backbone – – – – – – General Geometry — Transolver

Geometry Scaling – – – – – – – **Industrial-scale Mesh** — Transolver++, Trasolver-3

Physics Scaling – – – – – – – – – Large-scale Pre-training — GeoPT

“Magical Design” in Transolver





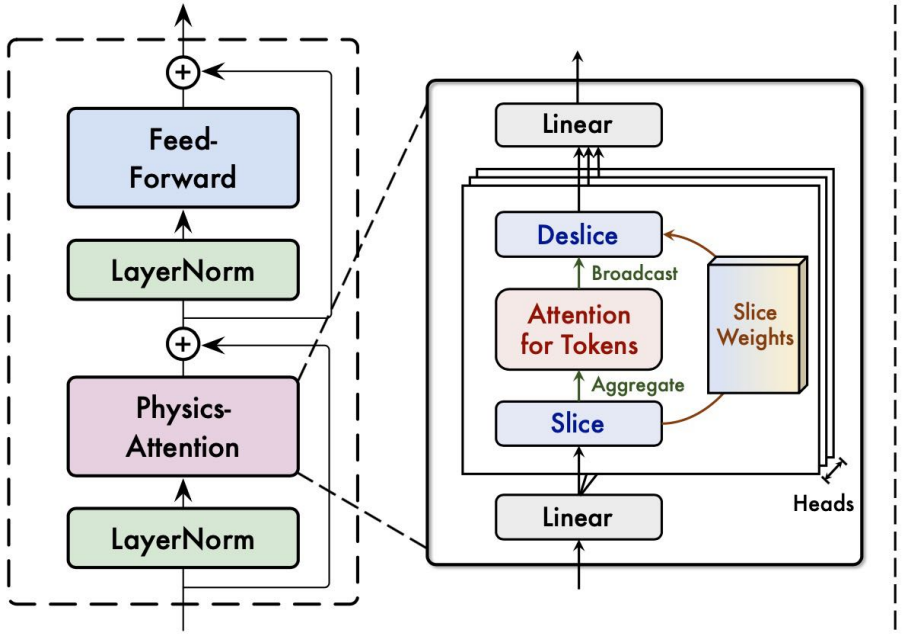
$$\mathbf{z}_j = \frac{\sum_{i=1}^{N} \mathbf{s}_{j,i}}{\sum_{i=1}^{N} \mathbf{w}_{i,j}} = \frac{\sum_{i=1}^{N} \mathbf{w}_{i,j}\mathbf{x}_i}{\sum_{i=1}^{N} \mathbf{w}_{i,j}}$$

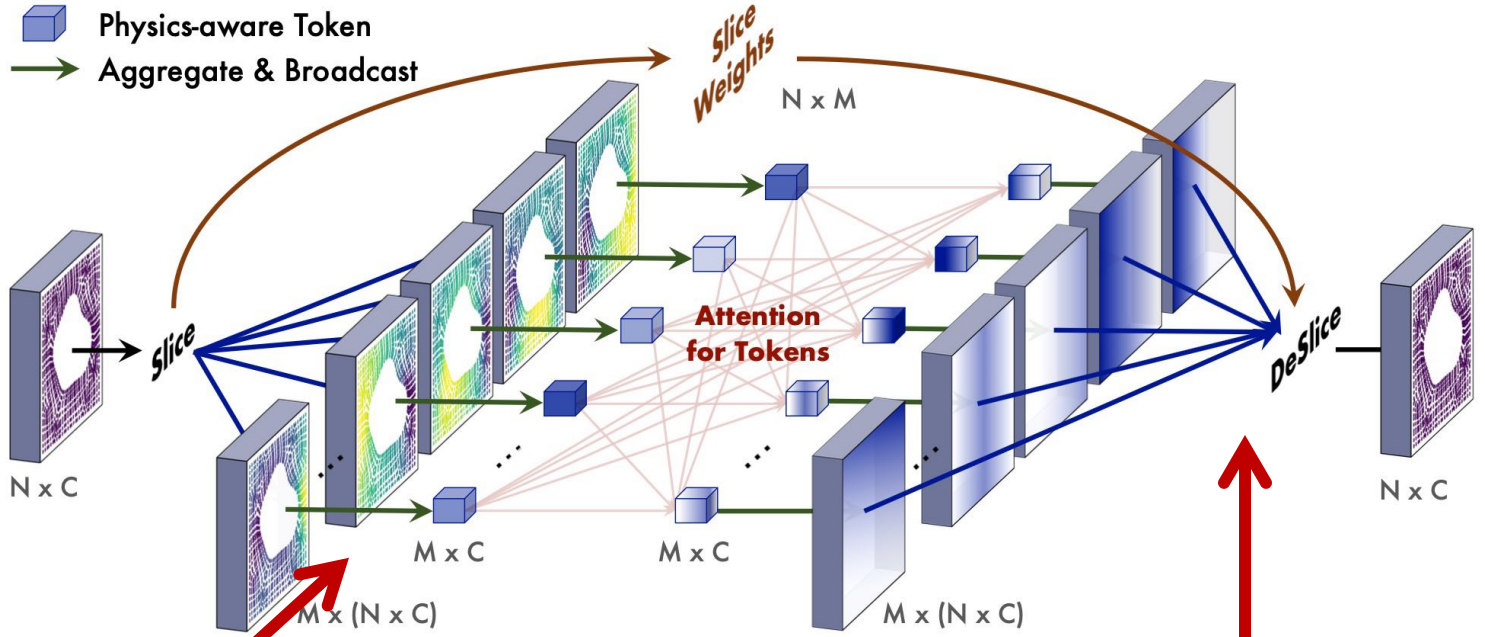
$$\mathbf{x}'_i = \sum_{j=1}^{M} \mathbf{w}_{i,j}\mathbf{z}'_j$$

Why adopt the global weighted sum?

Why reuse slice weights?

“Magical 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

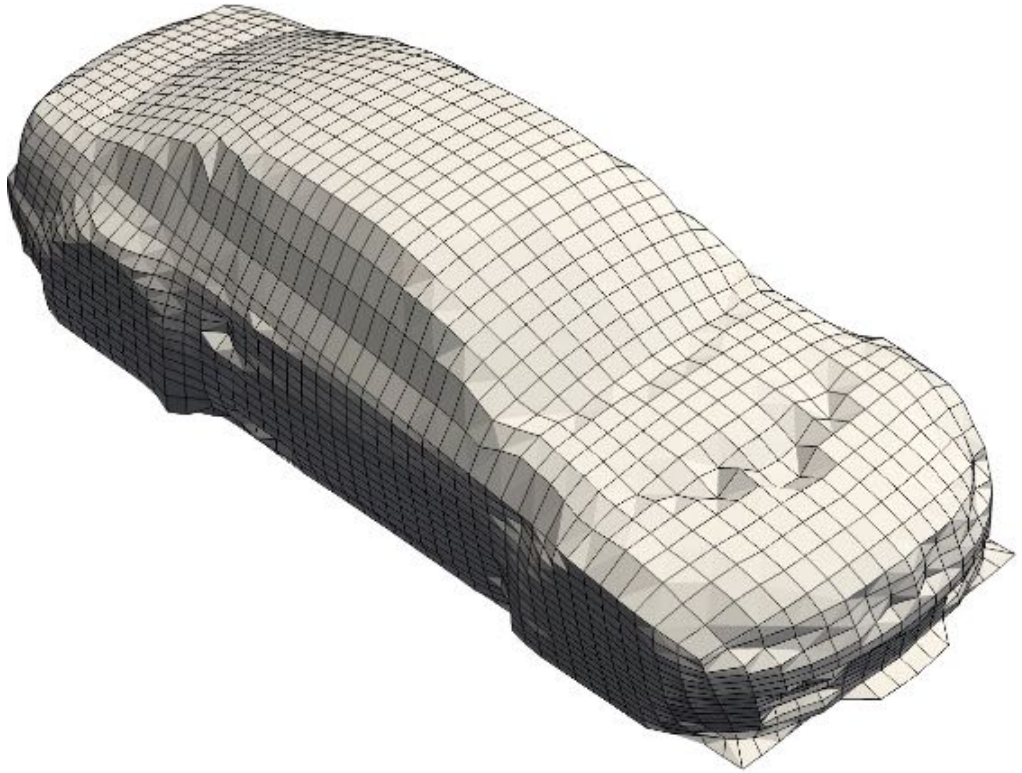
Yichen Di

Jianmin Wang

Mingsheng Long

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

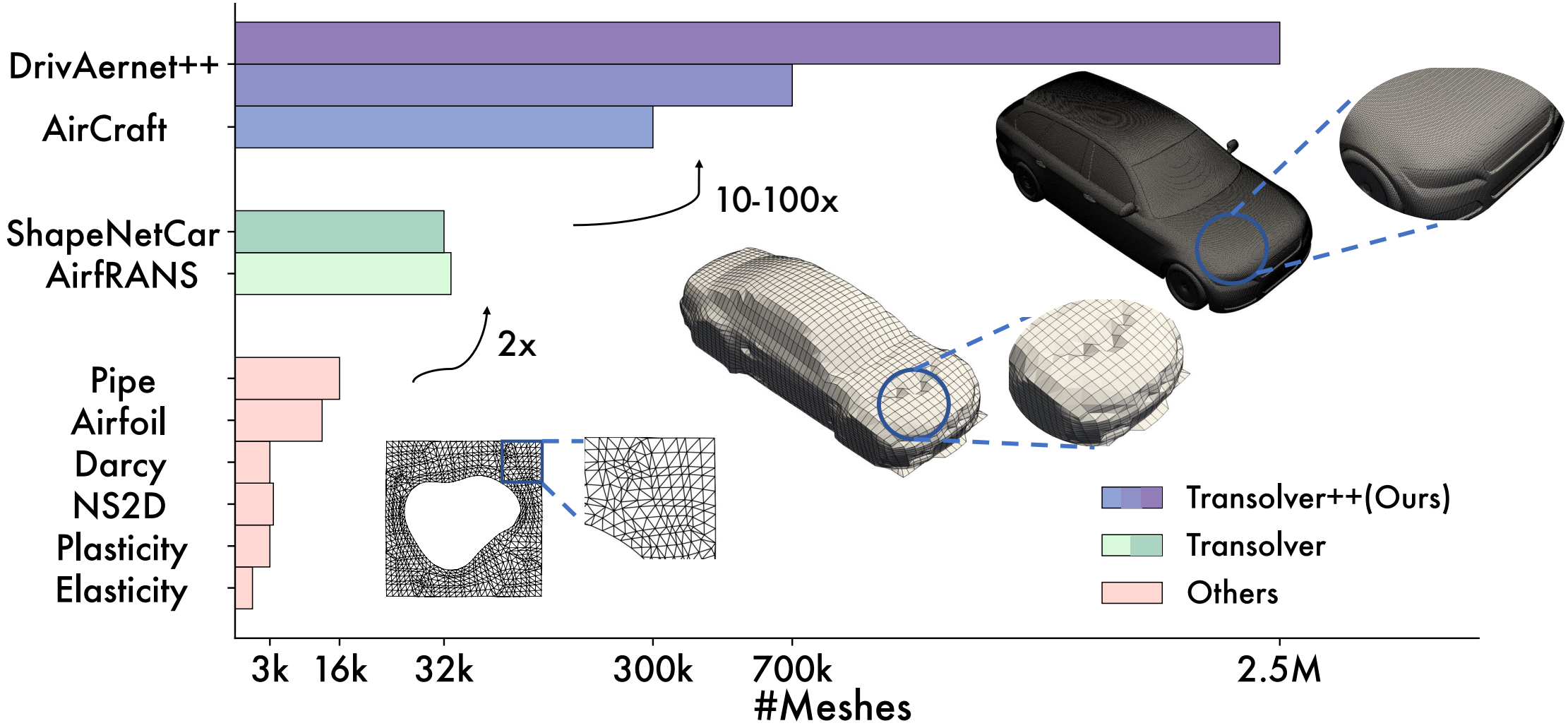
Extremely Large Geometries



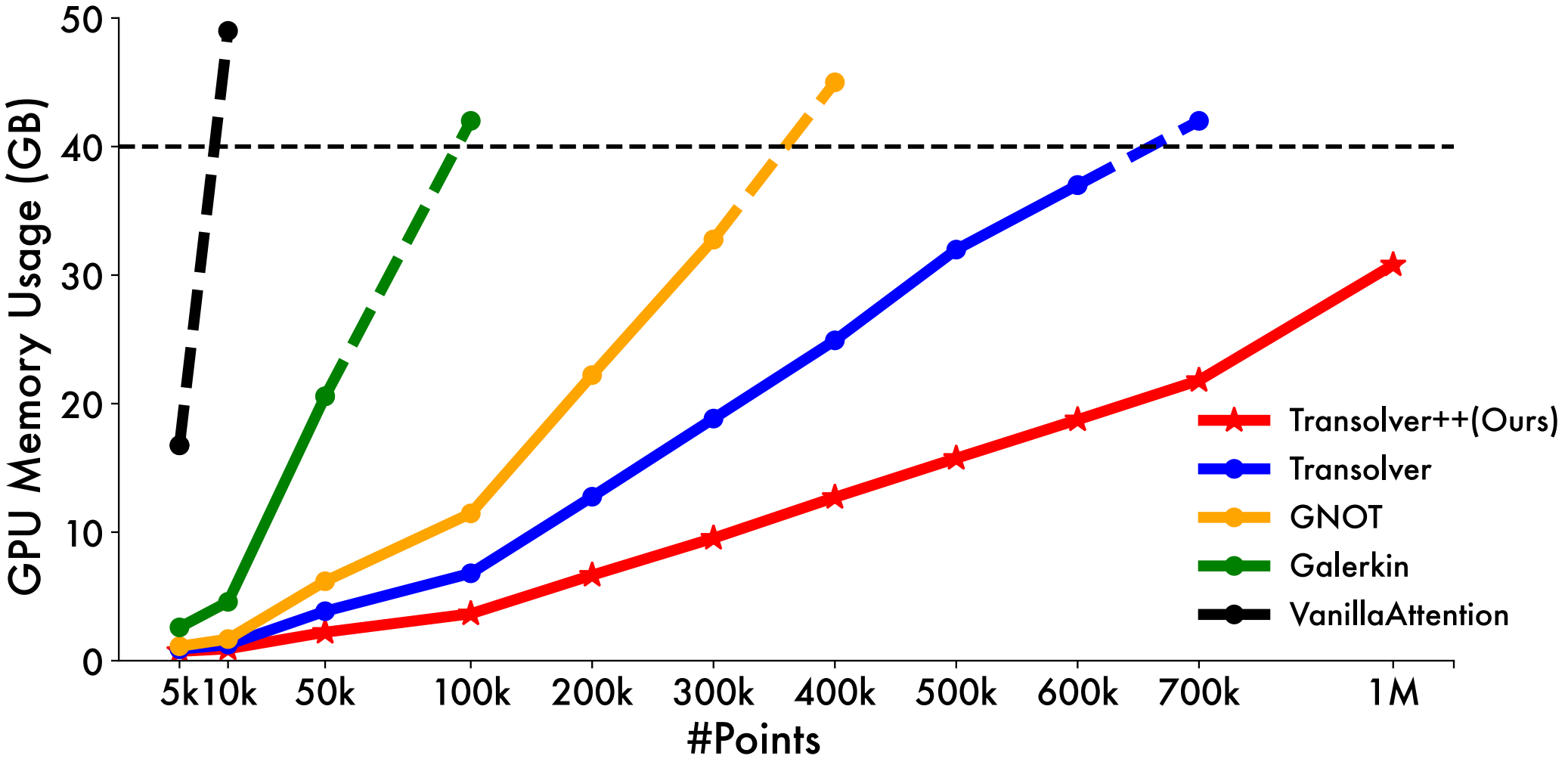
32k Mesh Points

2.5M Mesh Points

10-100x Larger than Previous Benchmarks

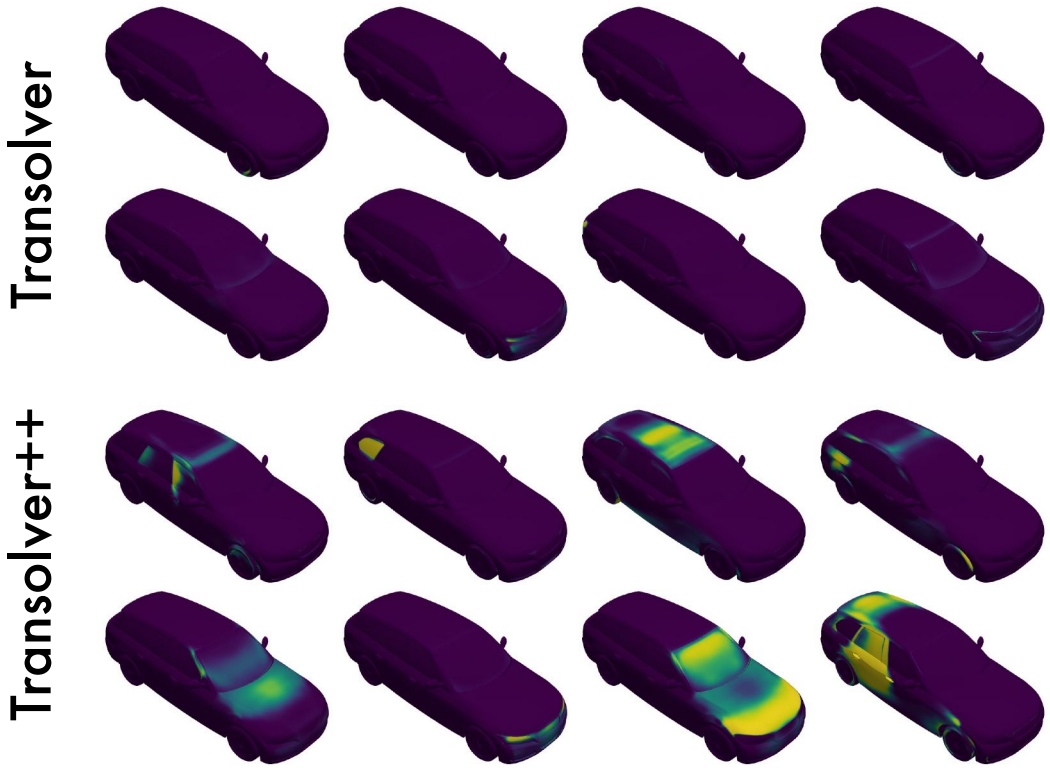


Transolver++: Enable Simulation in Million-Scale Geometries



Challenges within Transolver in Million-Scale Geometries

1. Homogeneous physical states



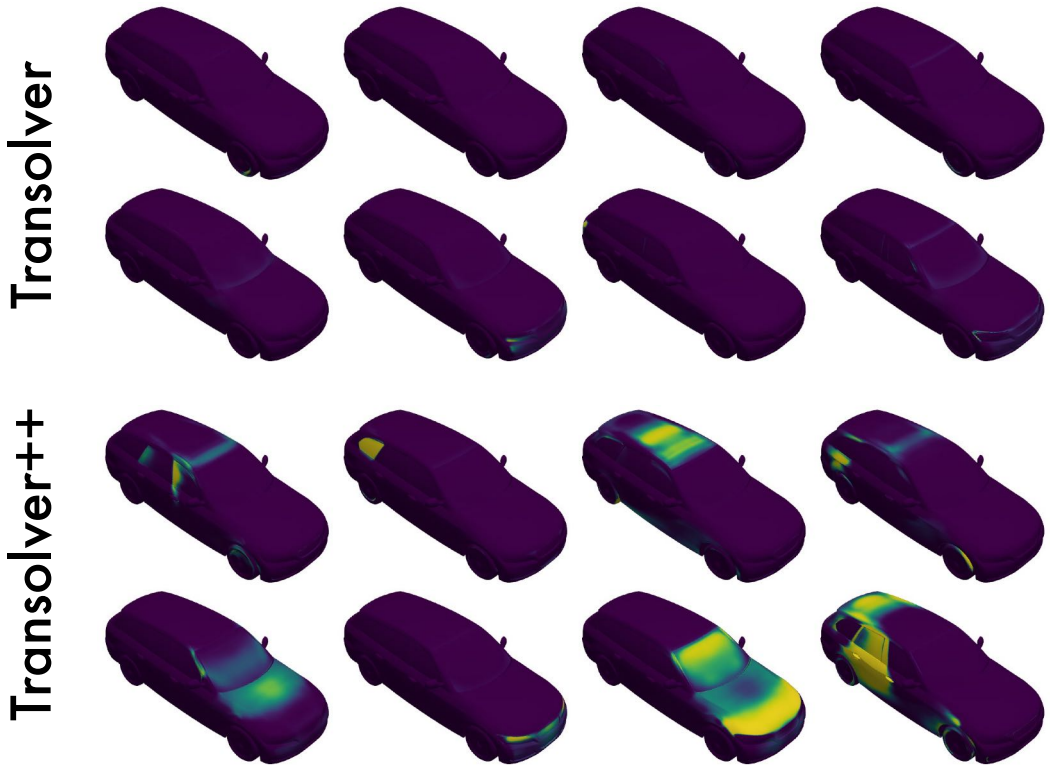
(b) Slice Weights Visualization

Degenerate in large-scale geometries

Improved physics learning

Challenges within Transolver in Million-Scale Geometries

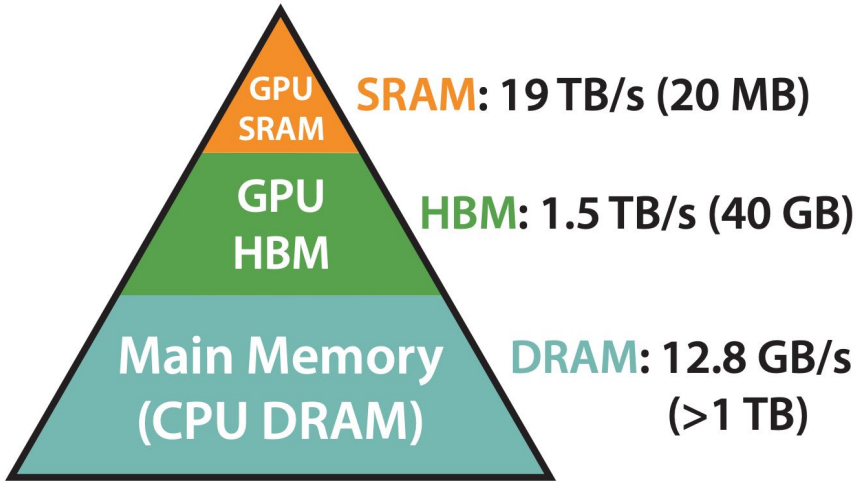
1. Homogeneous physical states



(b) Slice Weights Visualization

2. Efficiency Bottleneck

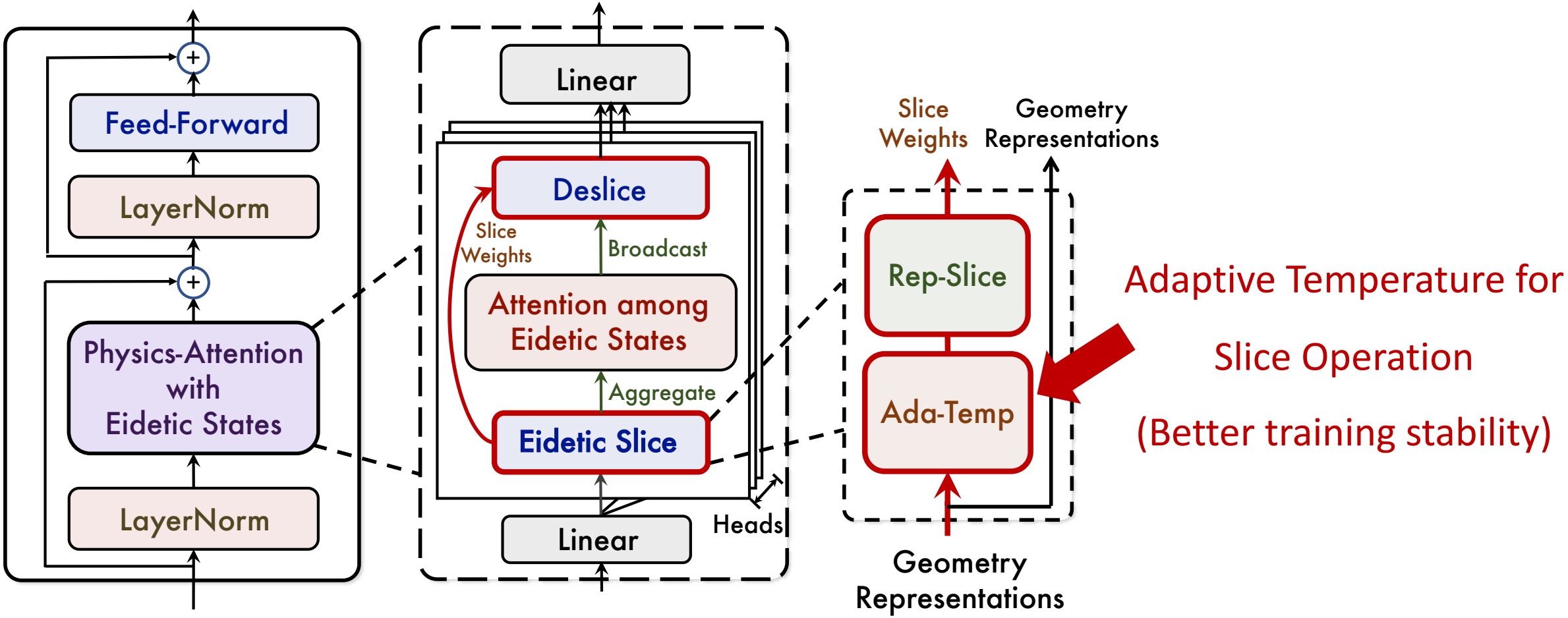
- Even a single intermediate representation of one million mesh points will consume **2GB of GPU memory**



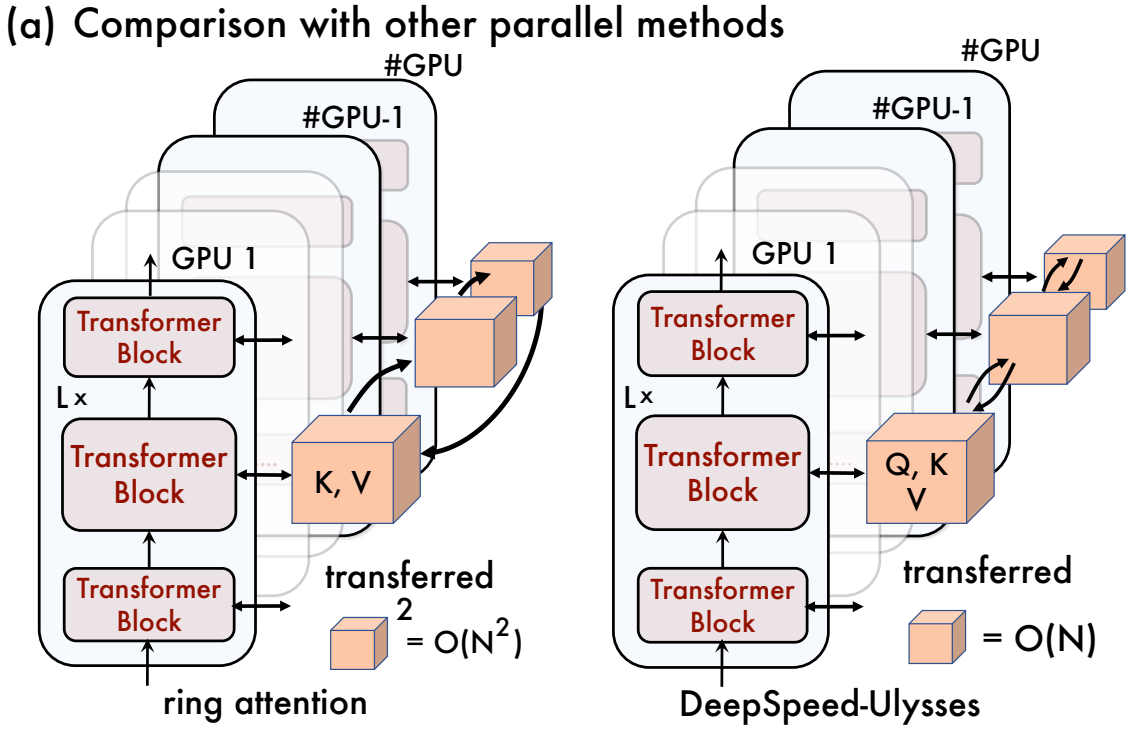
Memory Hierarchy with Bandwidth & Memory Size

Upgrade 1: Physics-Attention with Eidetic States

Architectural Design

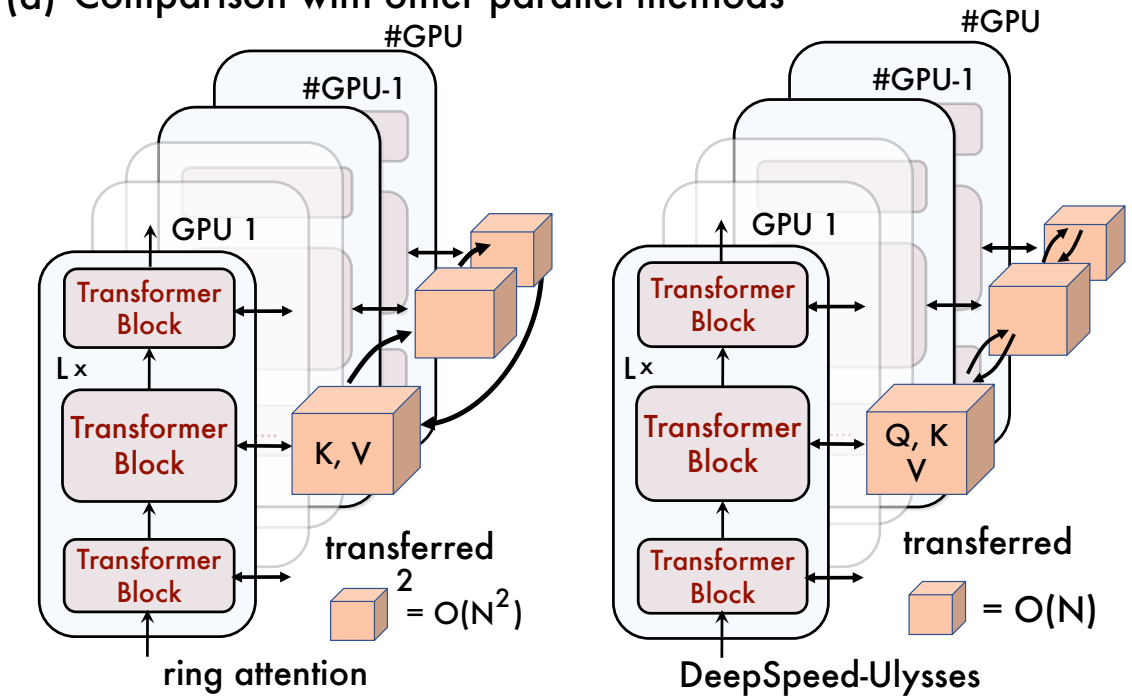


Upgrade 2: Parallelism Framework

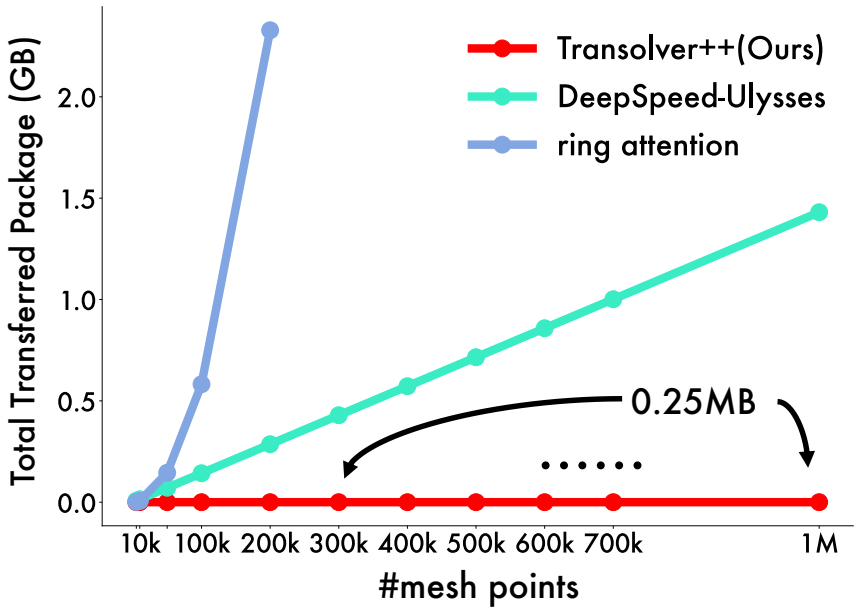


Upgrade 2: Parallelism Framework

(a) Comparison with other parallel methods



(b) Scalability of Transferred Package

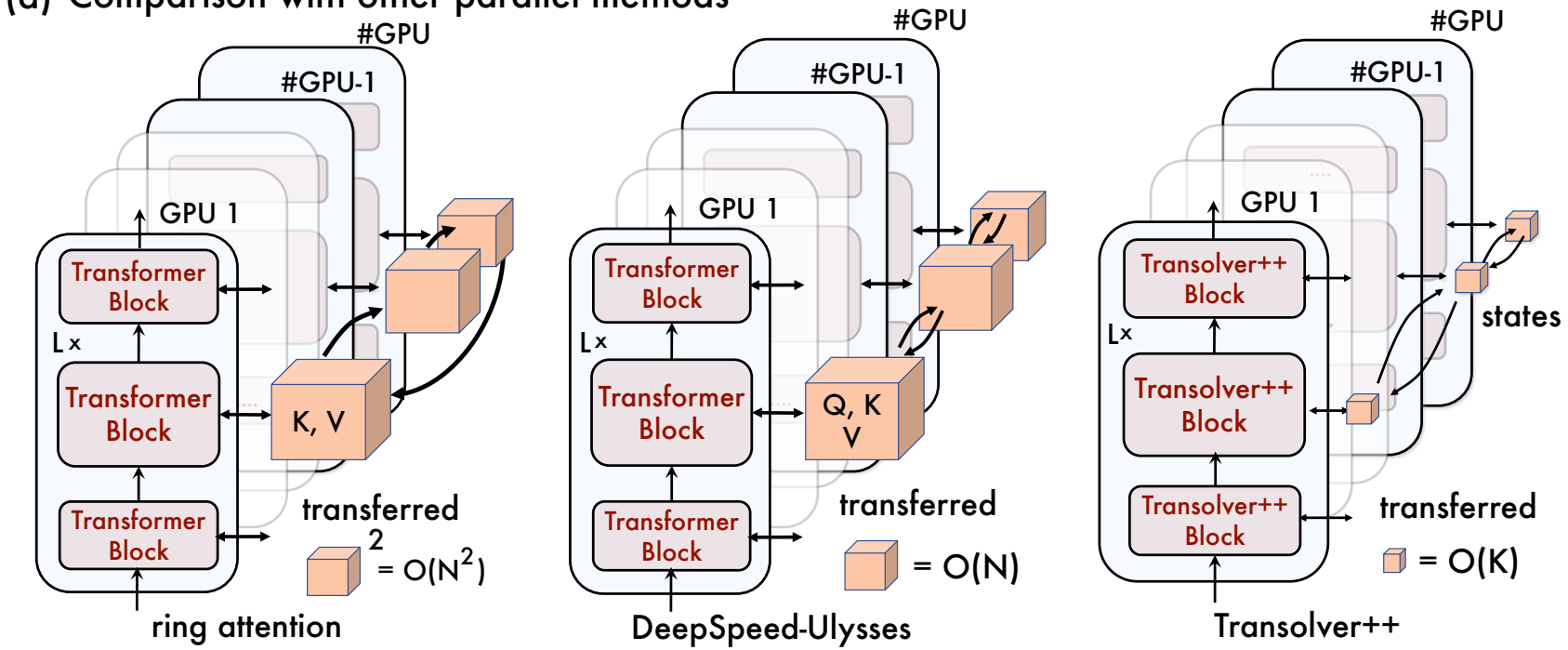


Upgrade 2: Parallelism Framework

Transolver is under a natively parallel formulation.

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

(a) Comparison with other parallel methods



Equivalent result

Accumulate **multi-GPU results**

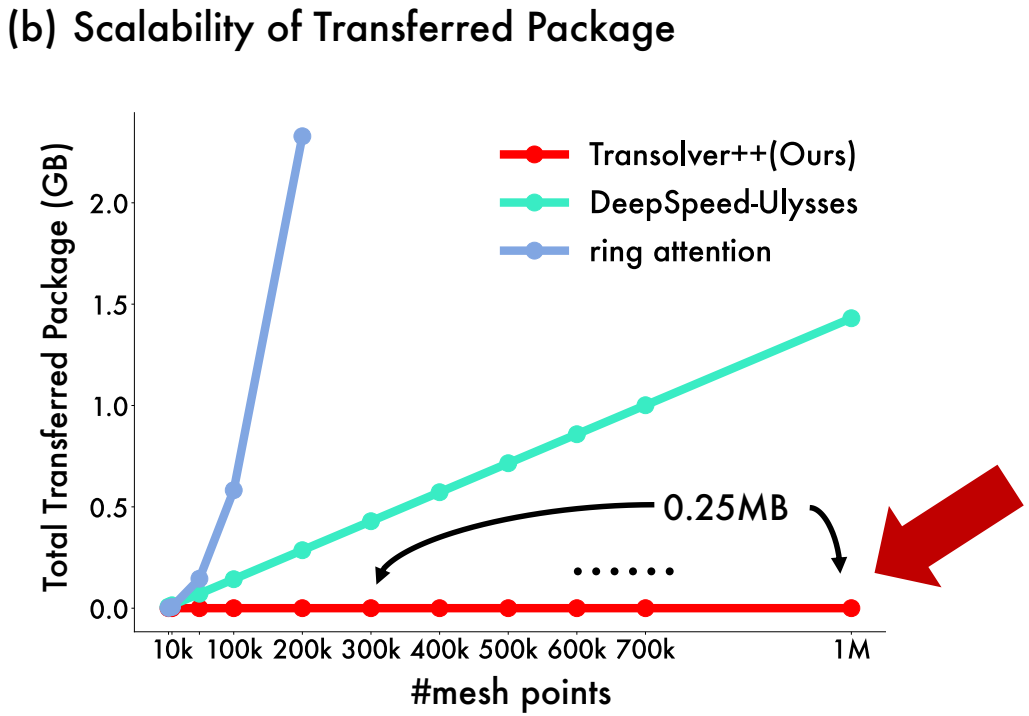
Compute physical states **in each GPU**

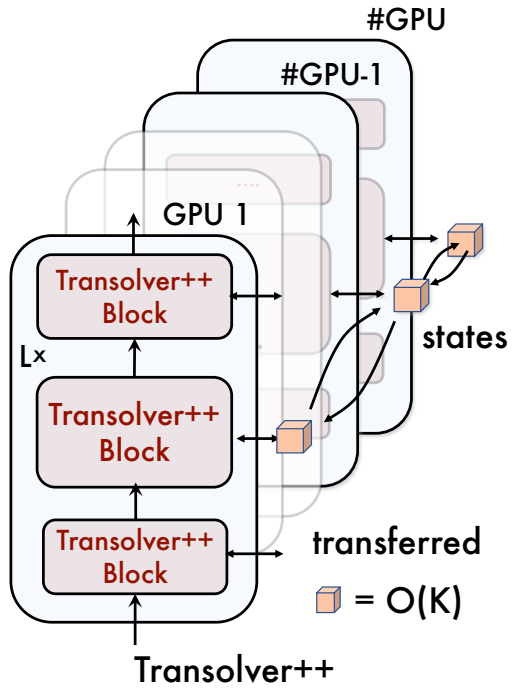
Split the mesh into **multiple GPUs**

Upgrade 2: Parallelism Framework

Transolver is under a natively parallel formulation.

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





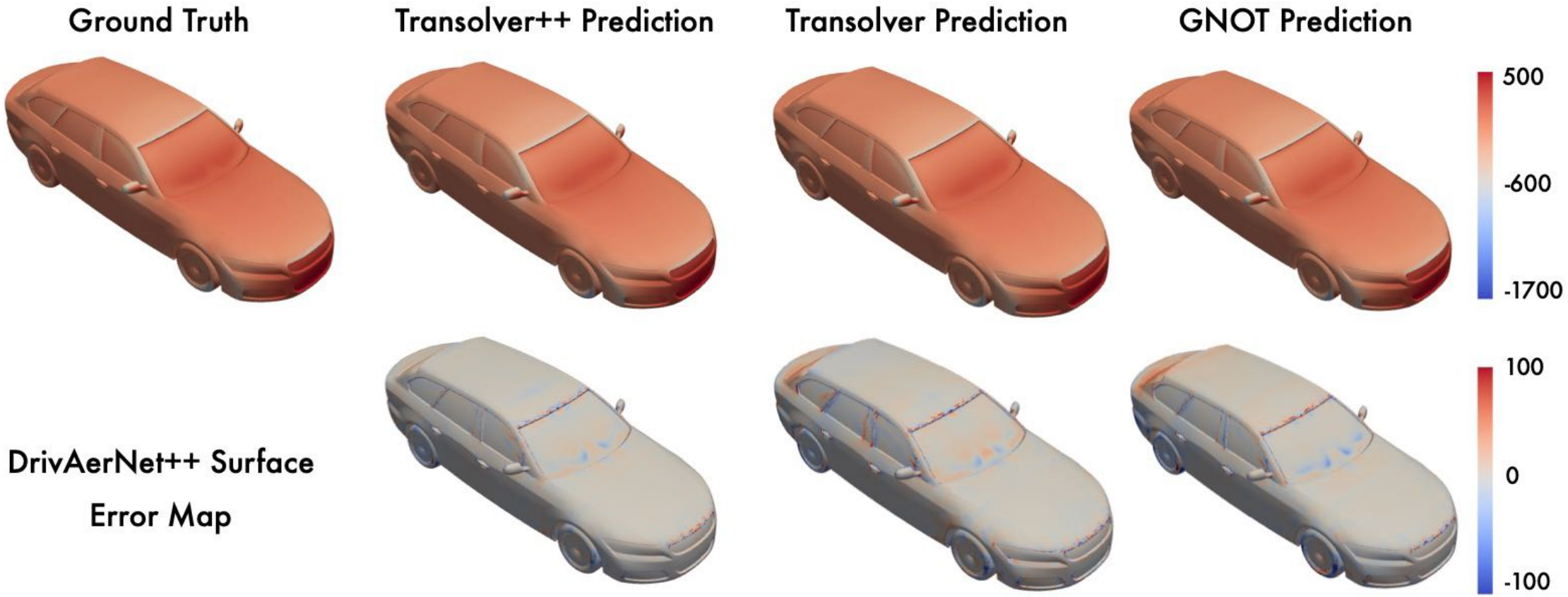
Equivalent result

Accumulate **multi-GPU results**

Compute physical states **in each GPU**

Split the mesh into **multiple GPUs**

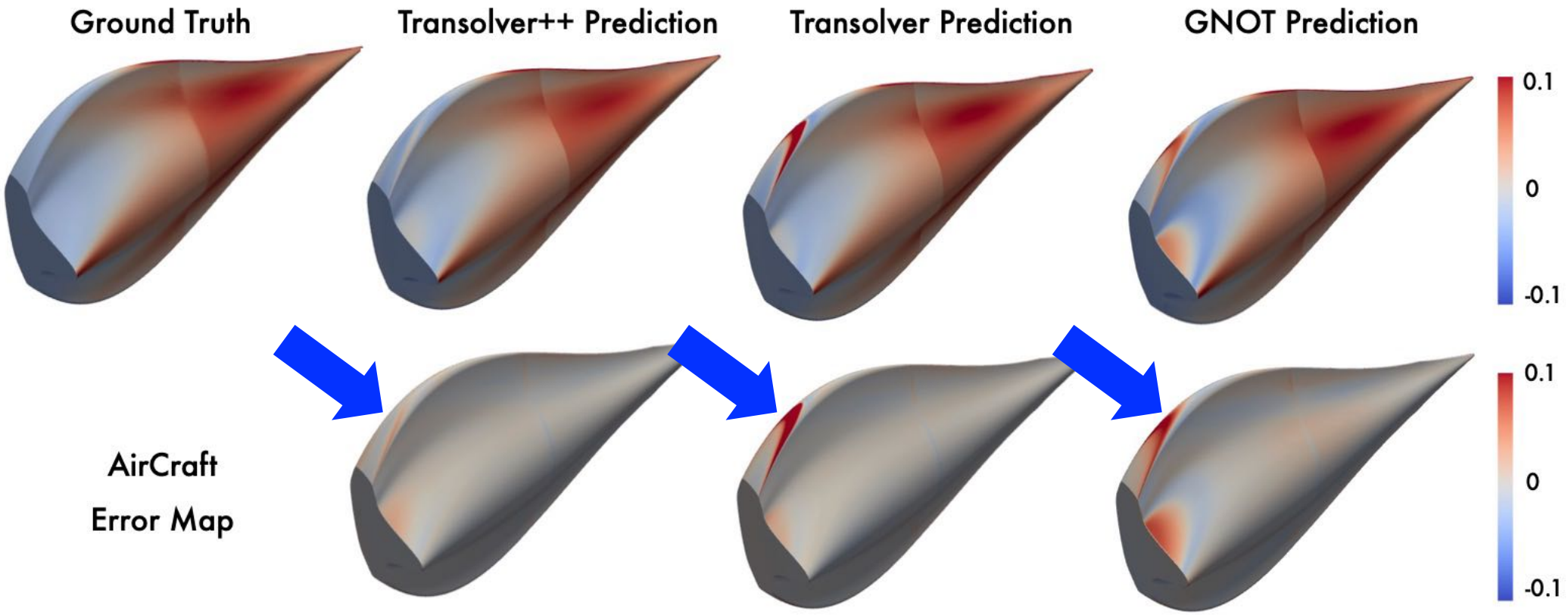
Industrial-level Applications: Car Design



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%.

ICML | 2026

The Forty-first International Conference on Machine Learning

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

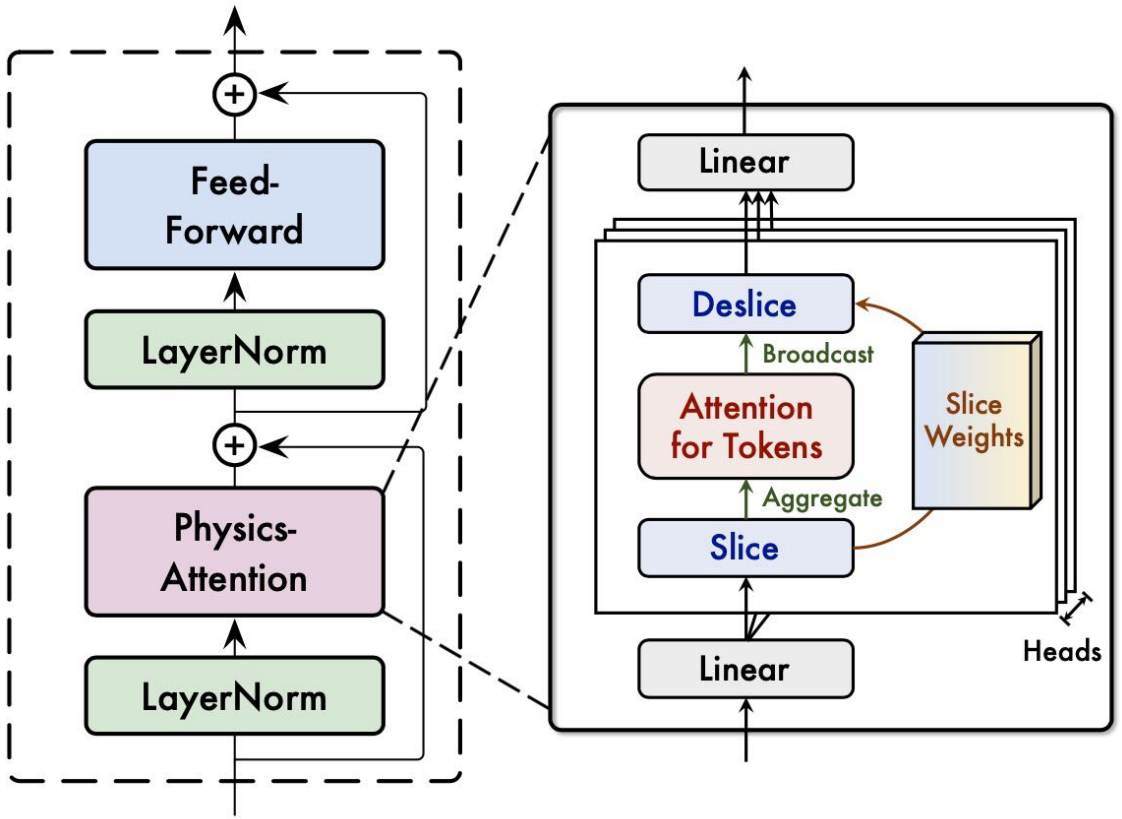
Yuezhou Ma

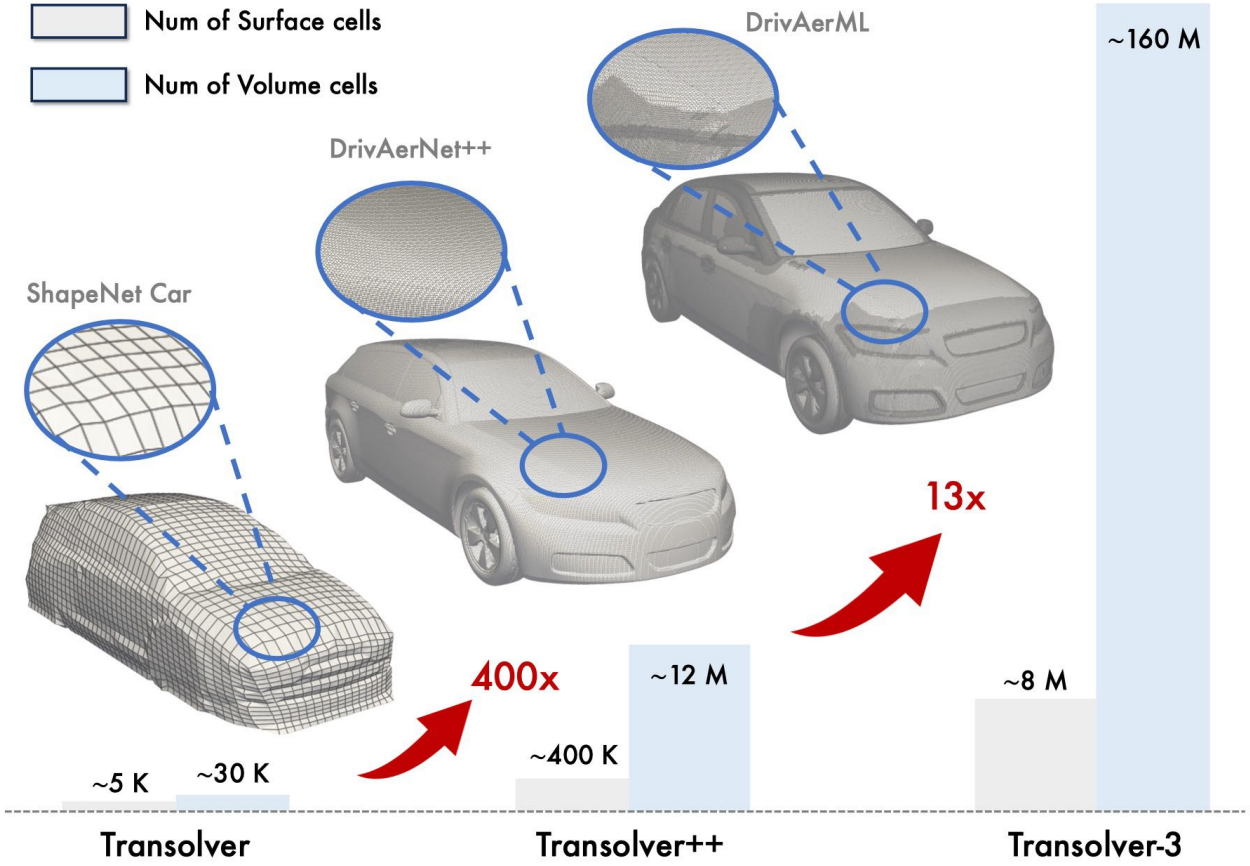
Huikun Weng

Jianmin Wang

Mingsheng Long

Scale to Over 100-Million-Cell Geometries





Detailed Complexity Analysis

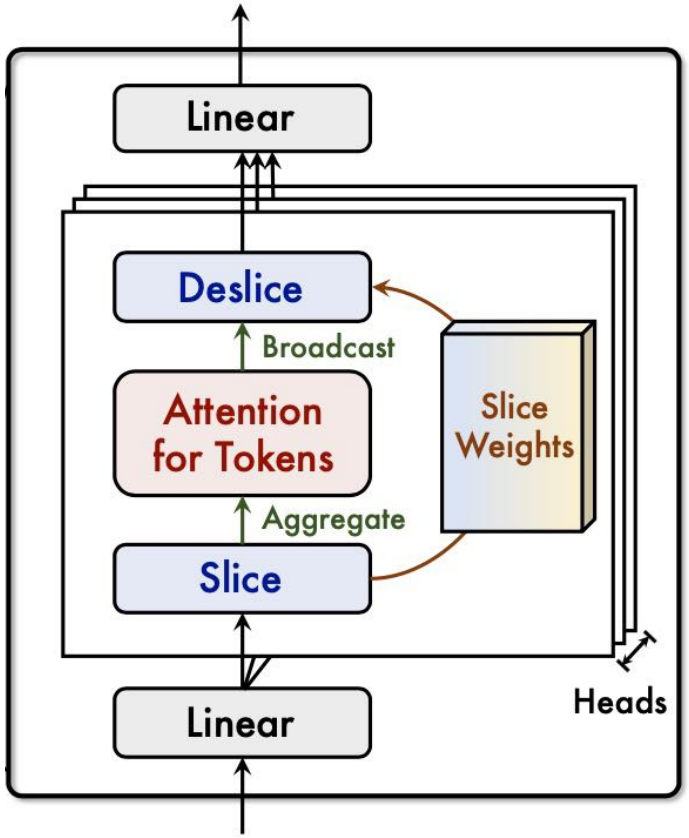


Table 1. Complexity Analysis of Original Physics-Attention.

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

Slice

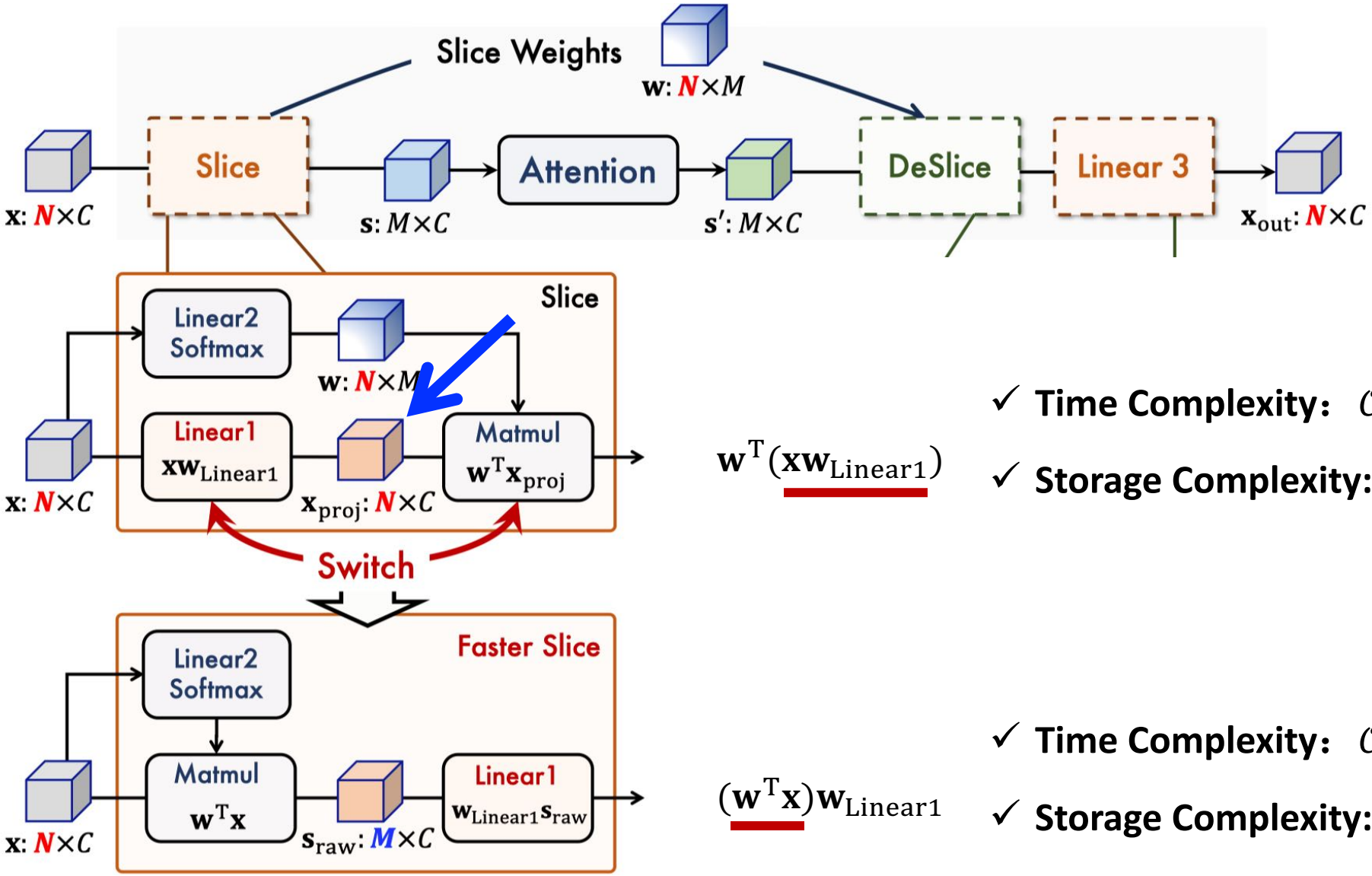
Attn

Deslice

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

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

Faster Slice



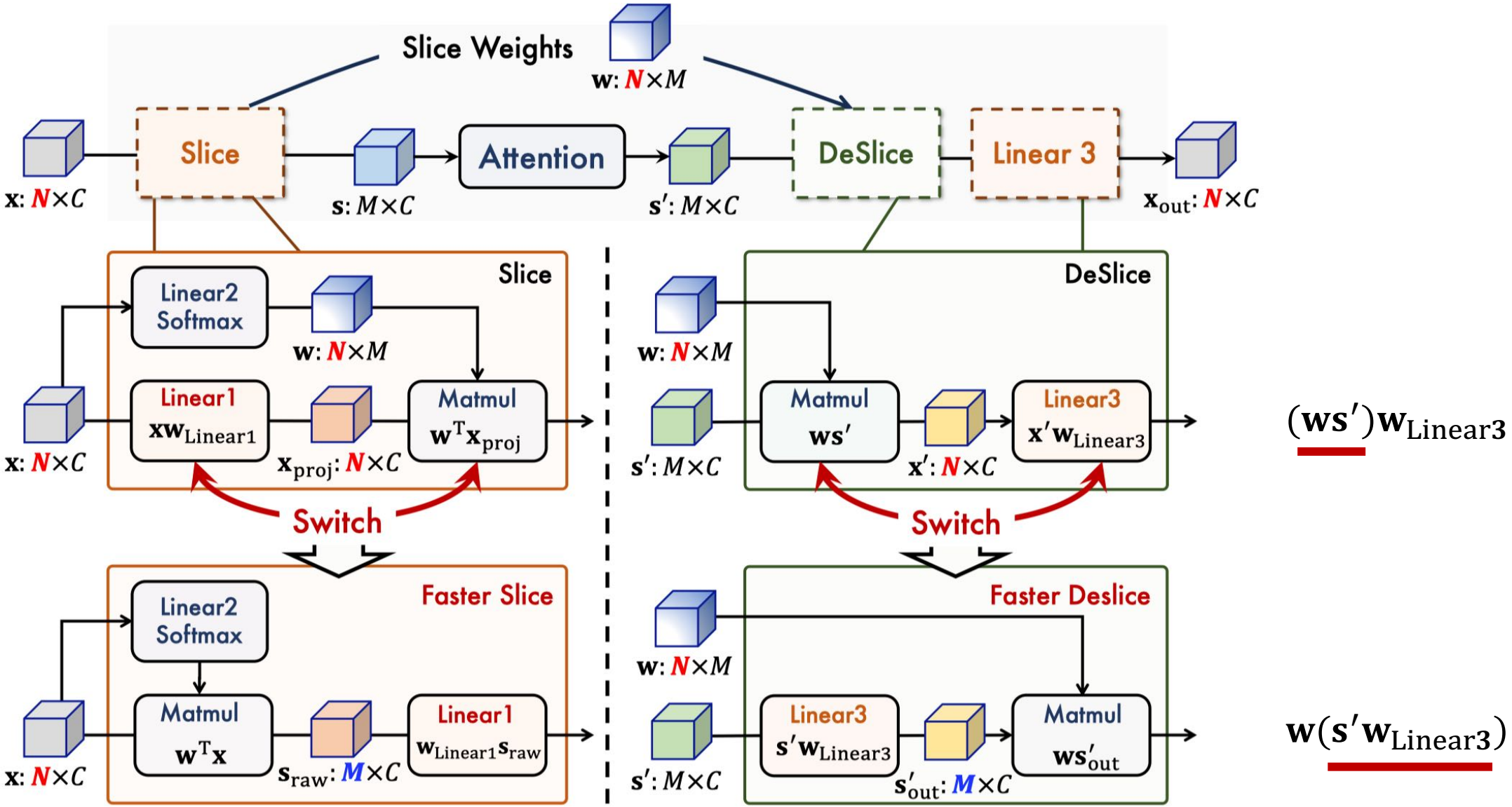
$\mathbf{w}^{\mathrm{T}}(\mathbf{x}\mathbf{w}_{\mathrm{Linear1}})$

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

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

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

Faster DeSlice

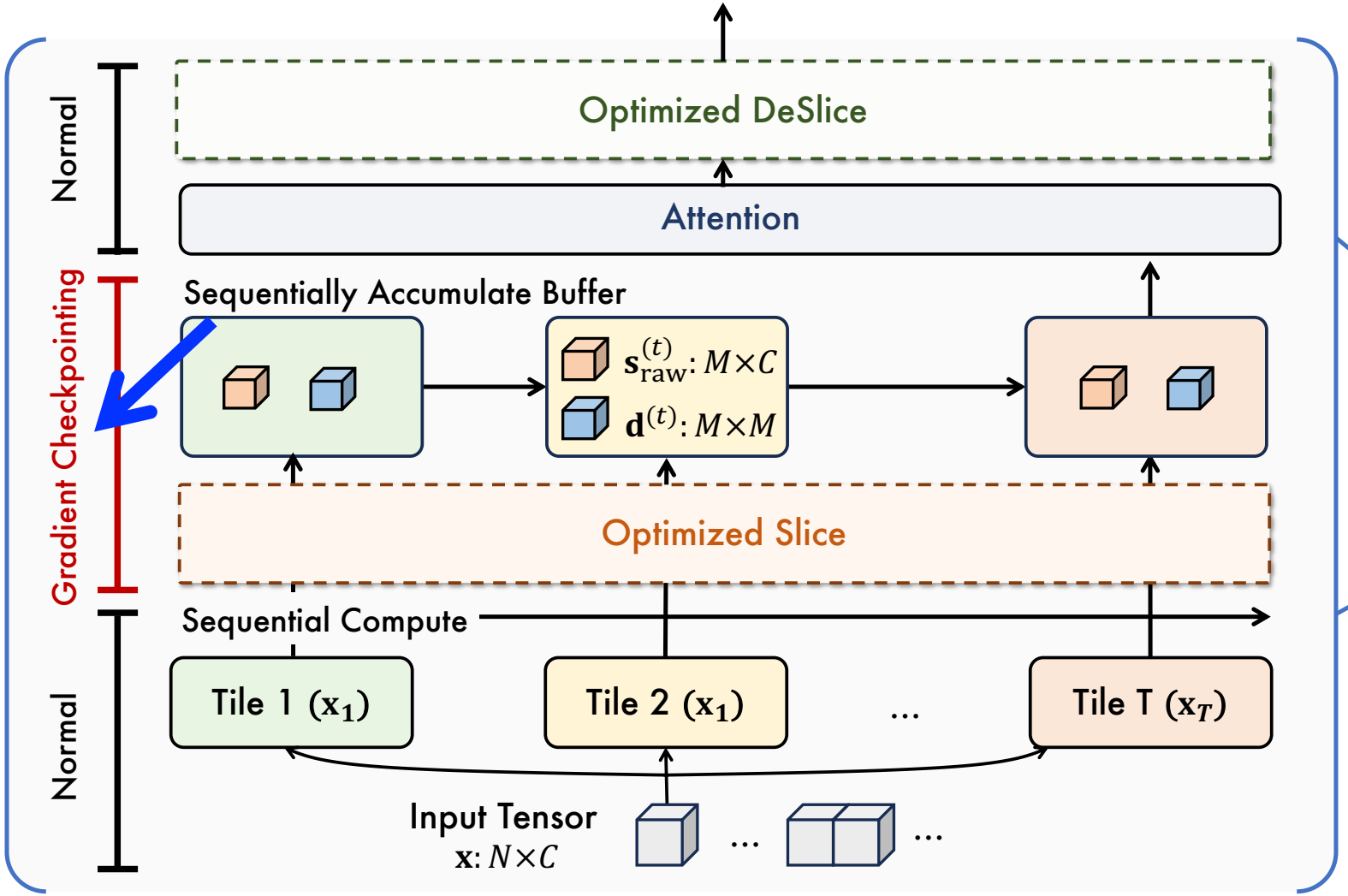


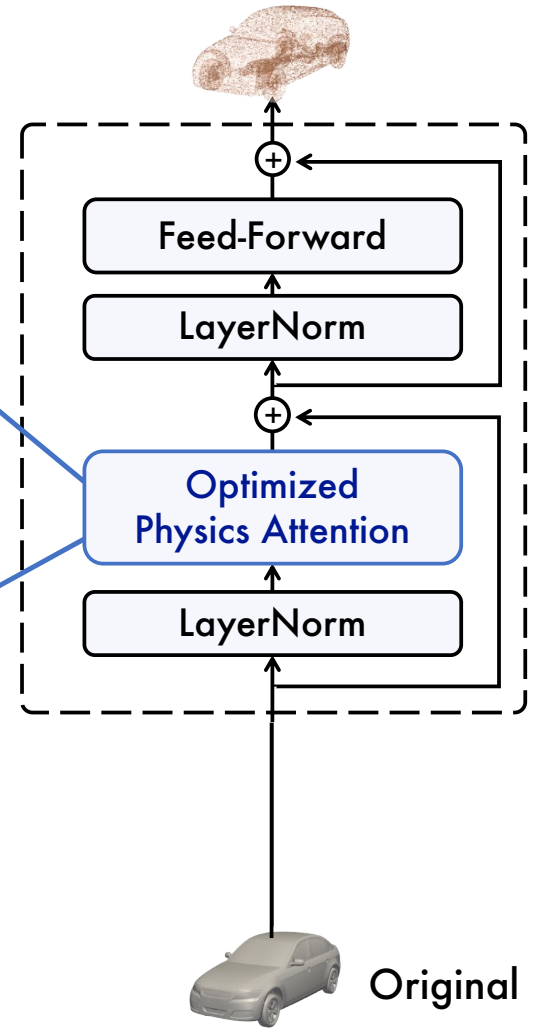
$(\mathbf{ws}')\mathbf{w}_{\mathrm{Linear3}}$

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

Training Scaling Framework

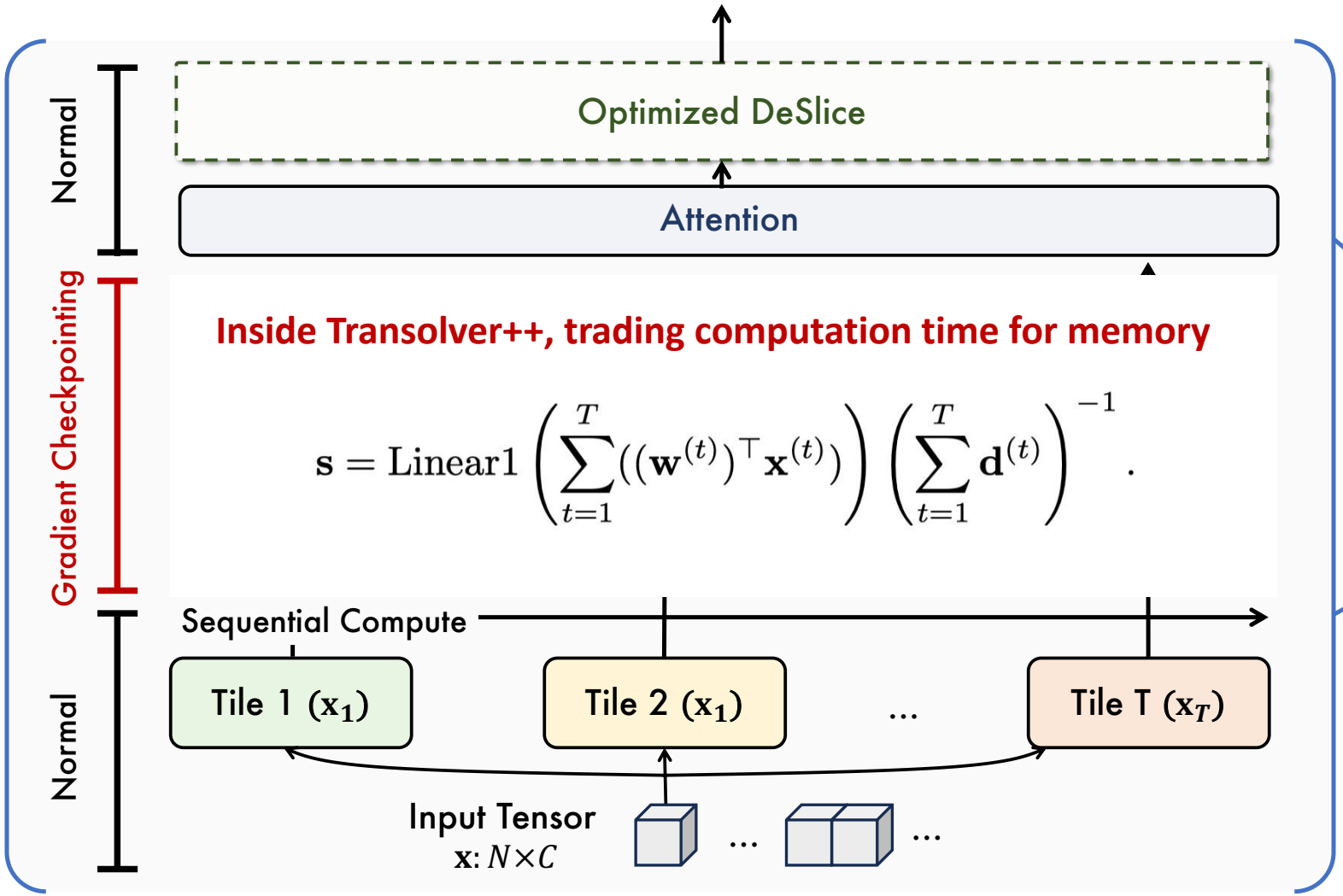
(a) Geometry Slice Tiling, reduce peaky memory usage

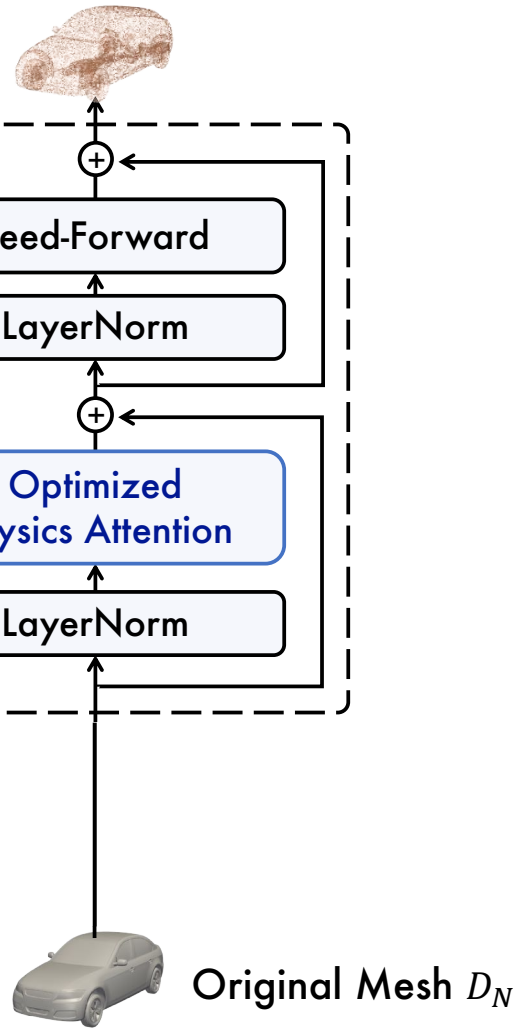




Training Scaling Framework

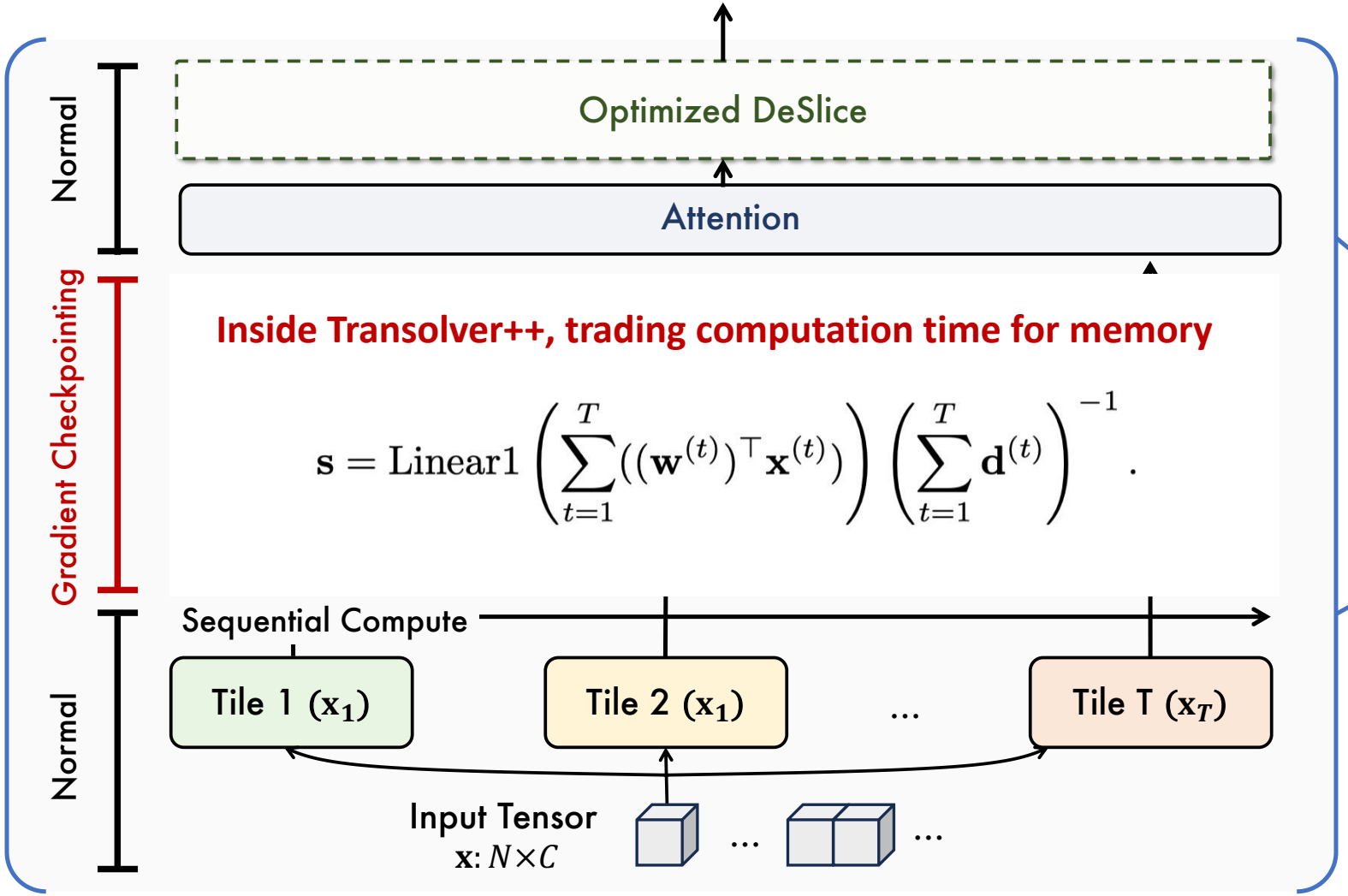
(a) Geometry Slice Tiling, reduce peaky memory usage



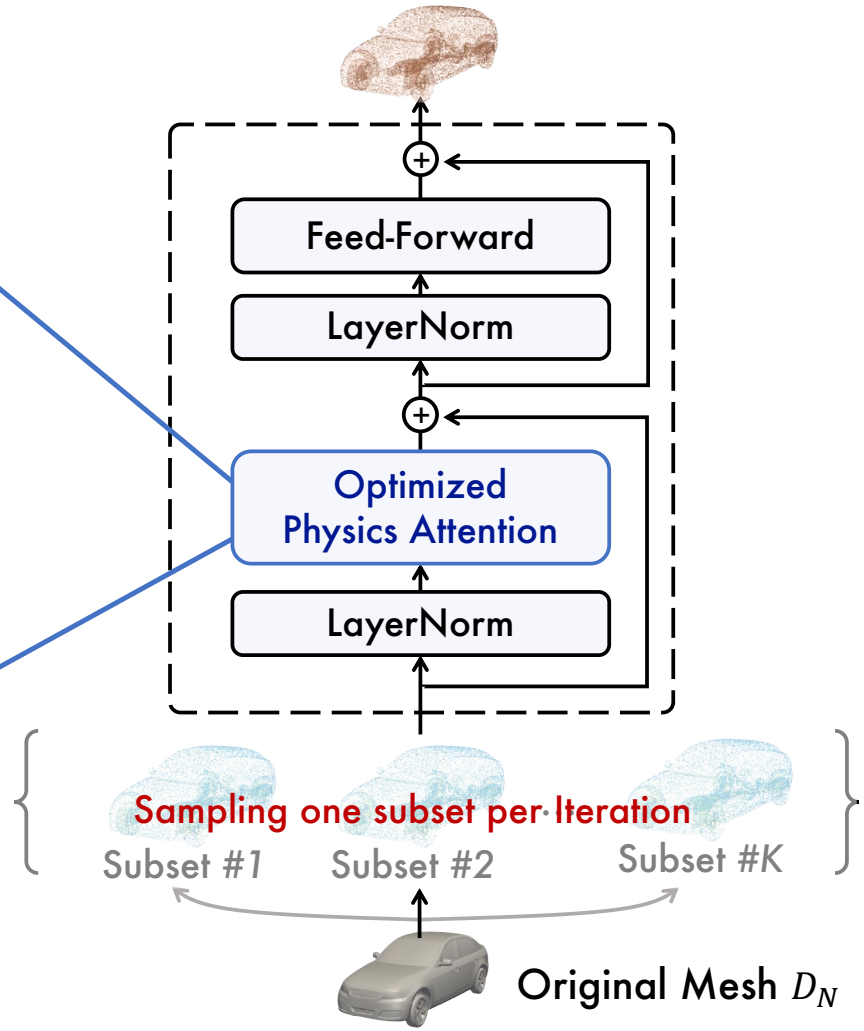


Training Scaling Framework

(a) Geometry Slice Tiling, reduce peaky memory usage

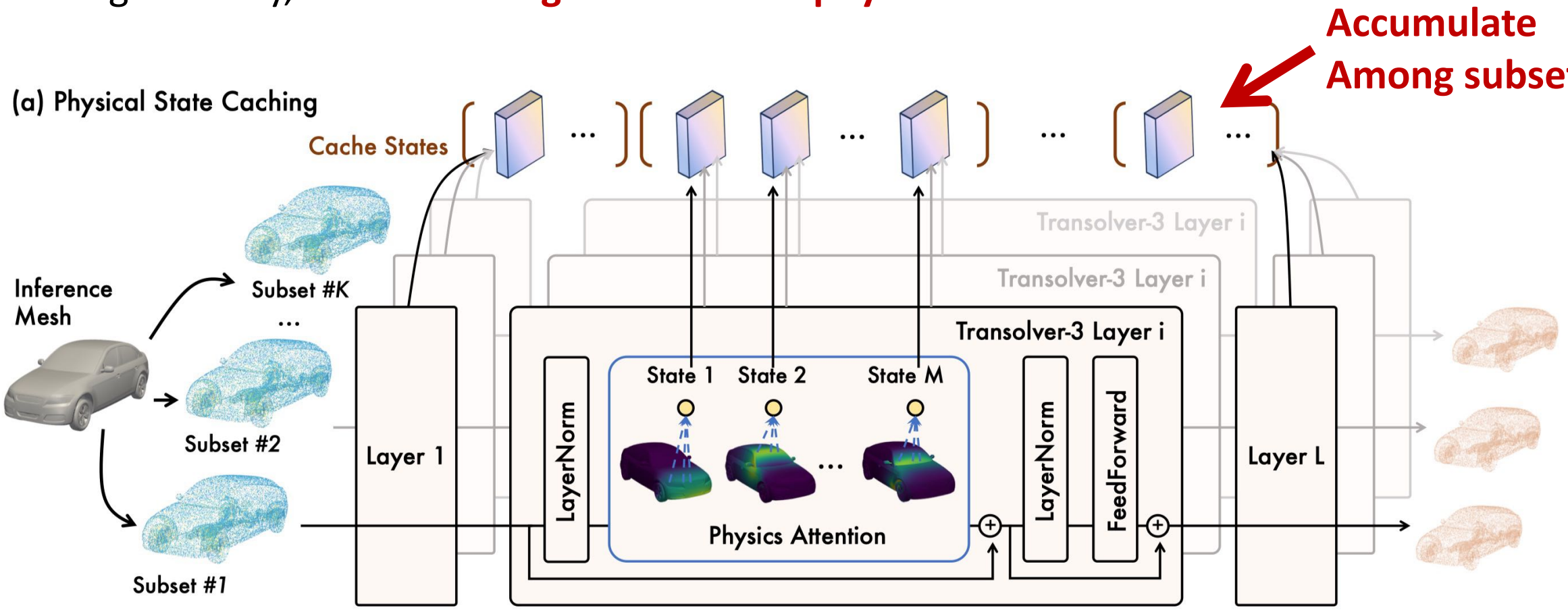


(b) Amortized Training



Inference Scaling Framework

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



Inference Scaling Framework

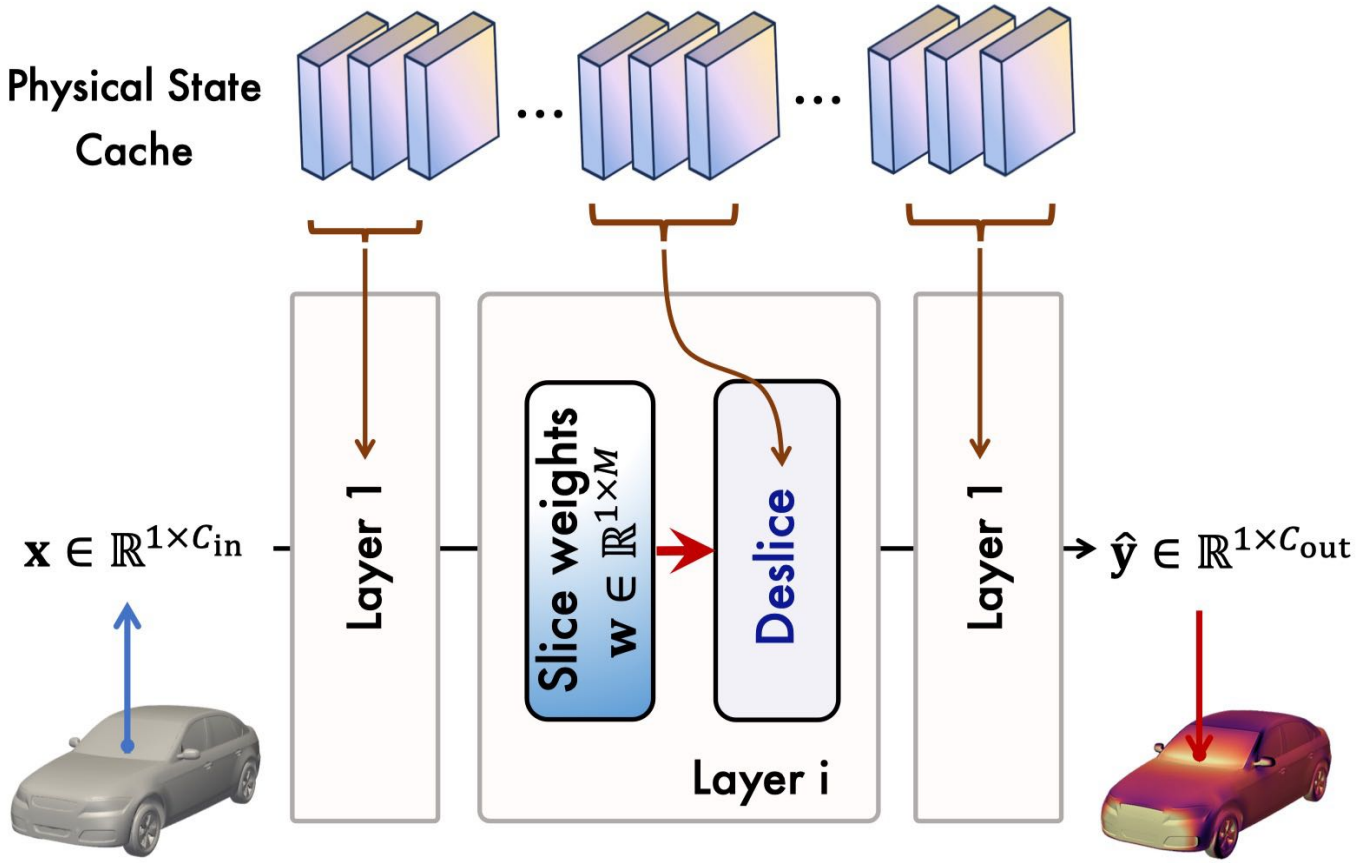
Inference on the **arbitrary position (in PINN style).**

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

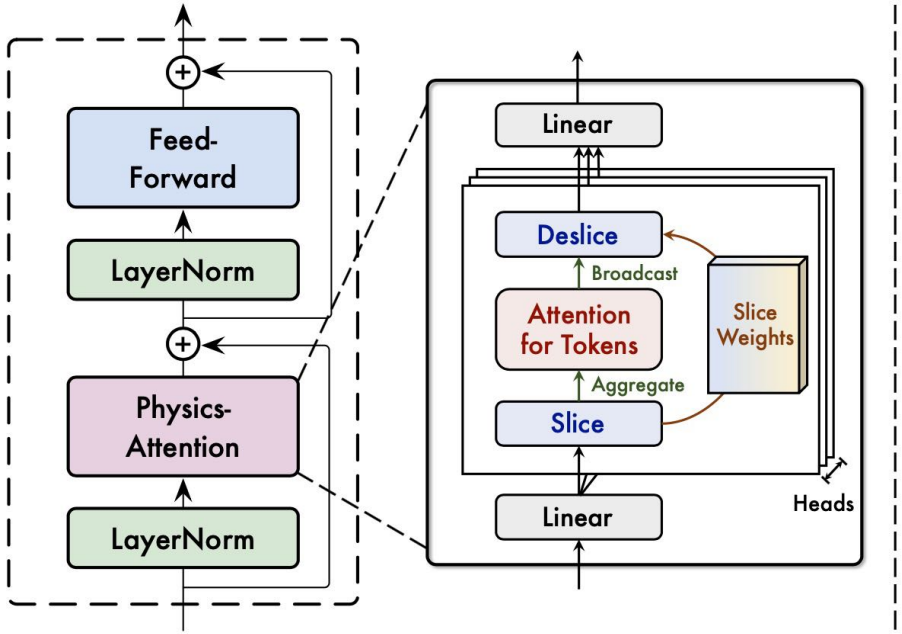
$$\mathbf{x}_{\mathbf{out}}^{(l)} = \mathbf{w}^{(l)}\mathbf{s}_{\mathbf{out}}^{\prime(l)}$$

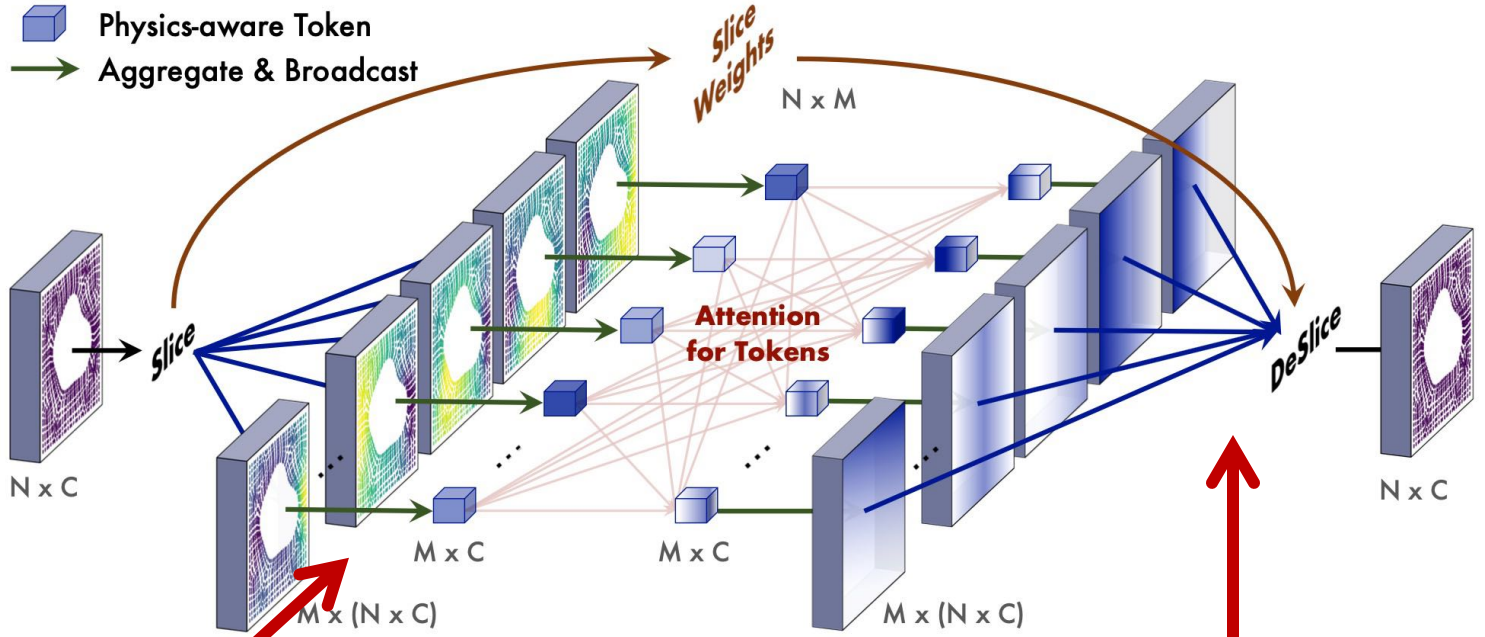
Cached physical states

Newly estimated slice weights



“Magical Design” in Transolver





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

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

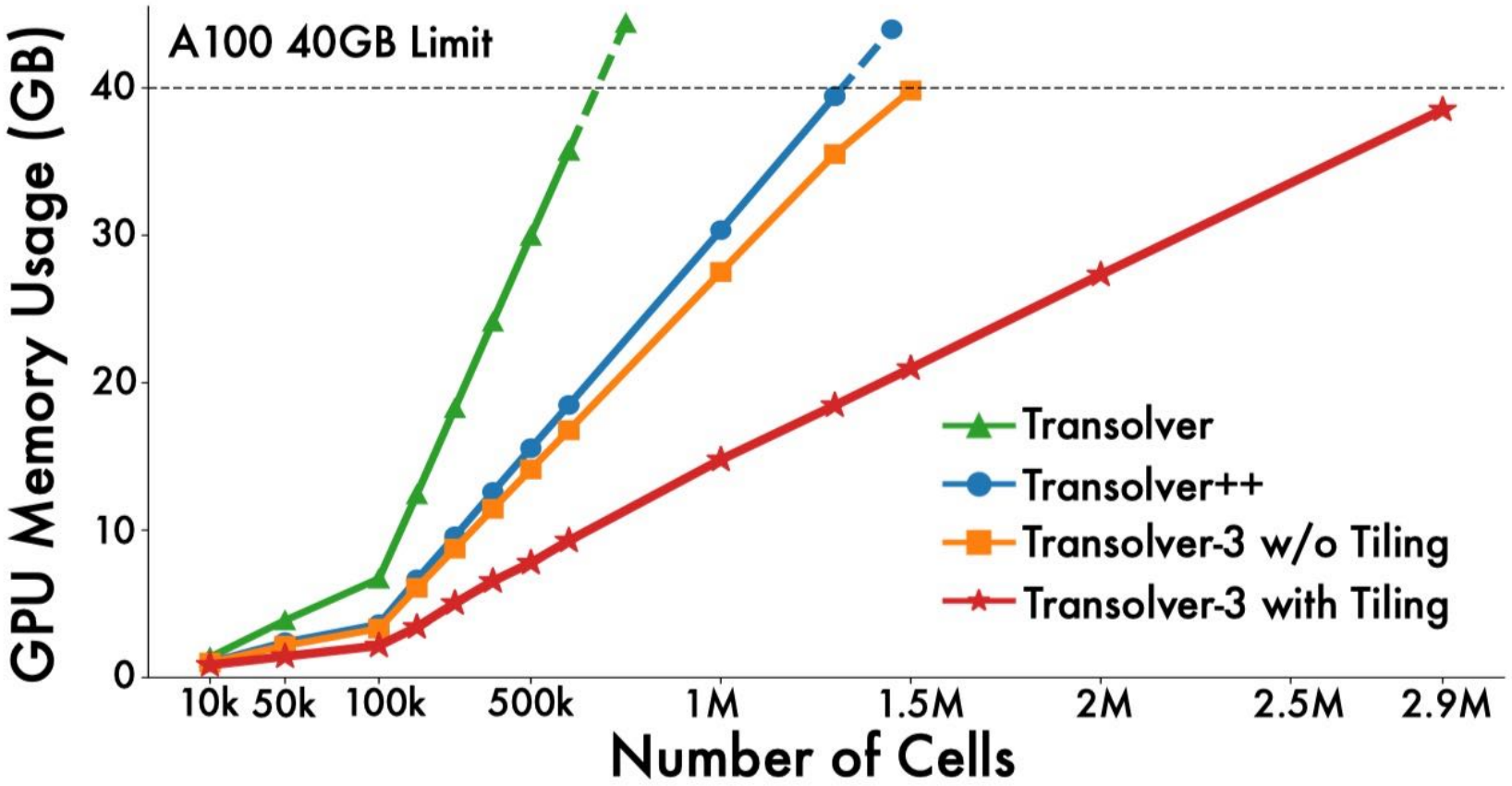
Why adopt the global weighted sum?

Support Transolver++

Why reuse slice weights?

Support Transolver-3

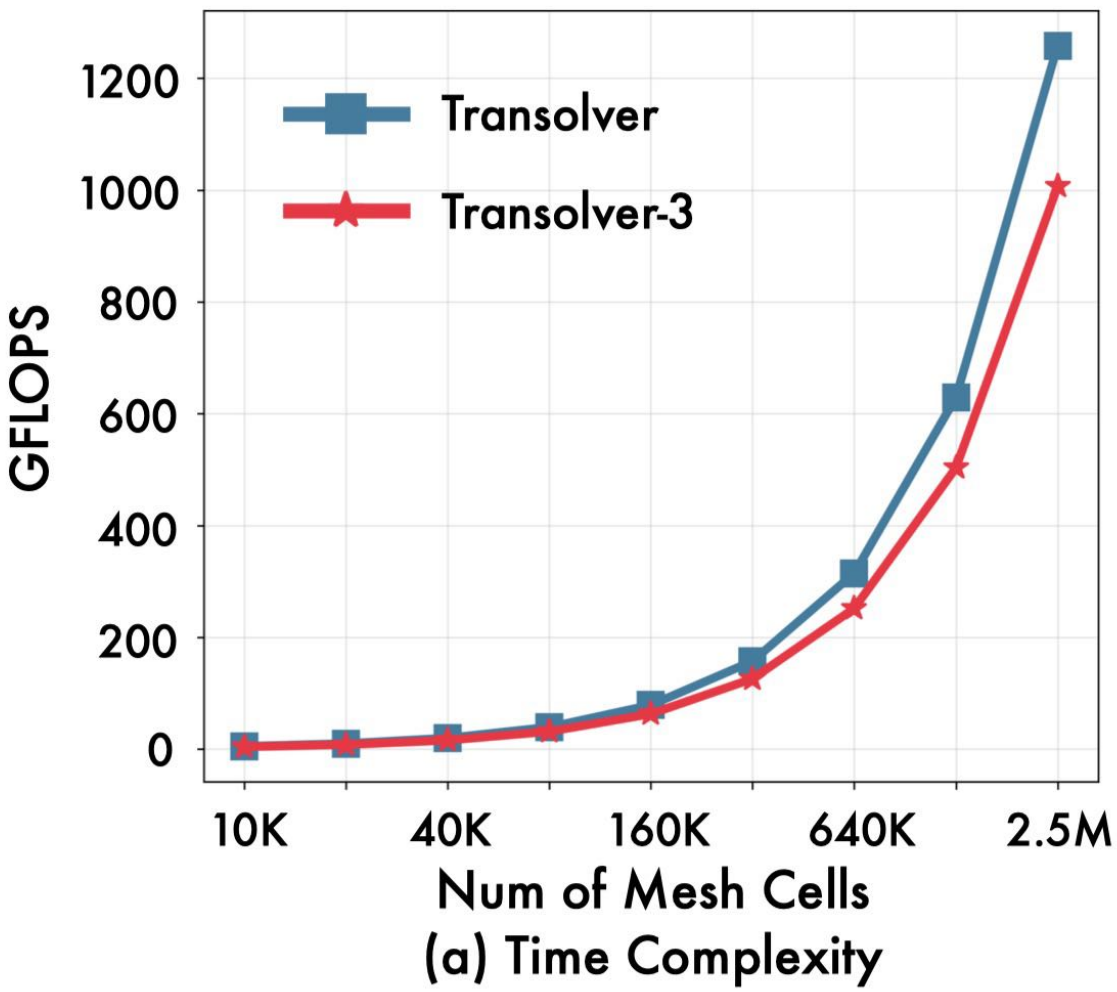
Efficiency Analysis (Geometry Scaling)

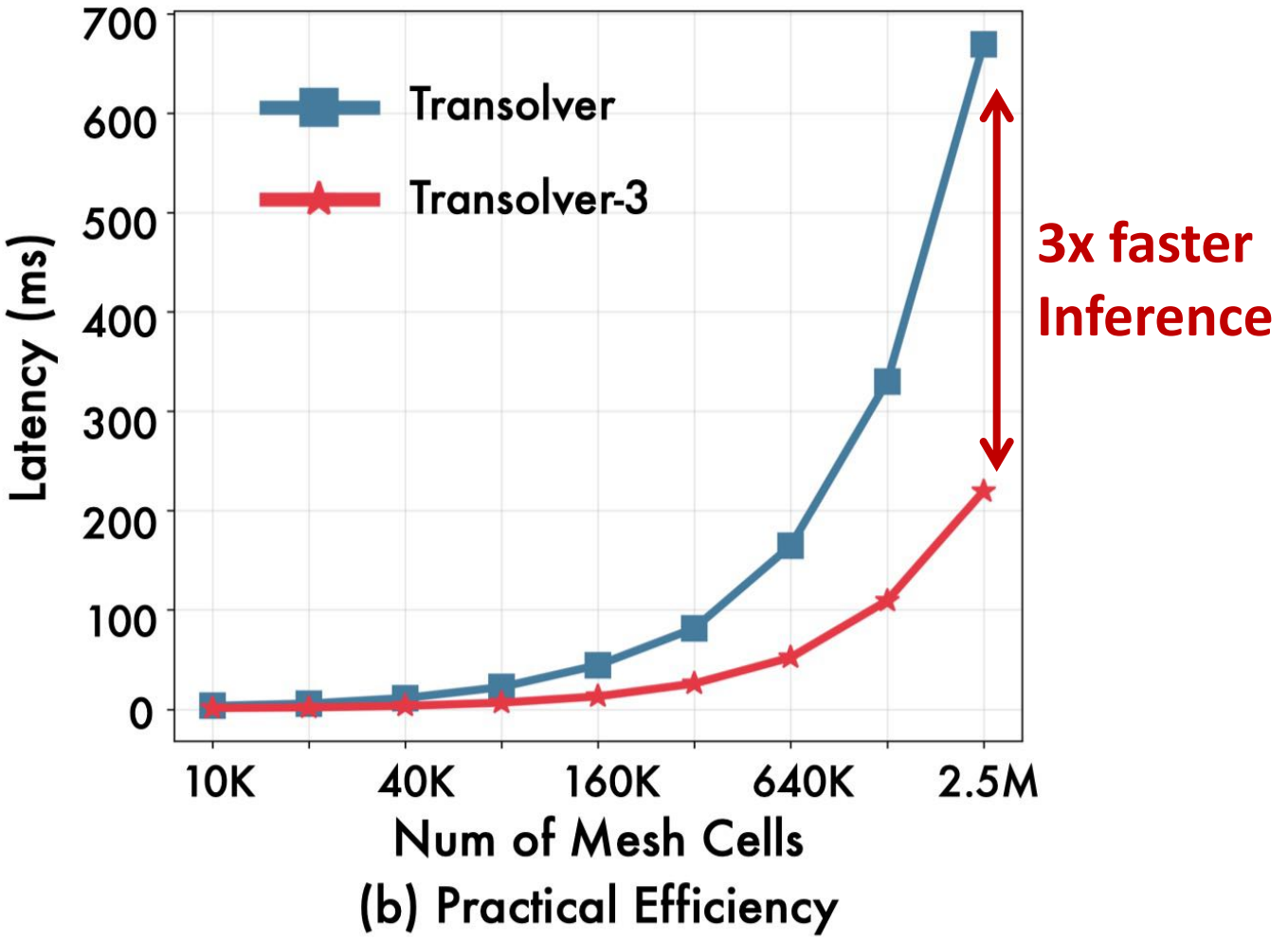


With slice tiling, Transolver-3 can process around 3M points on a single GPU.

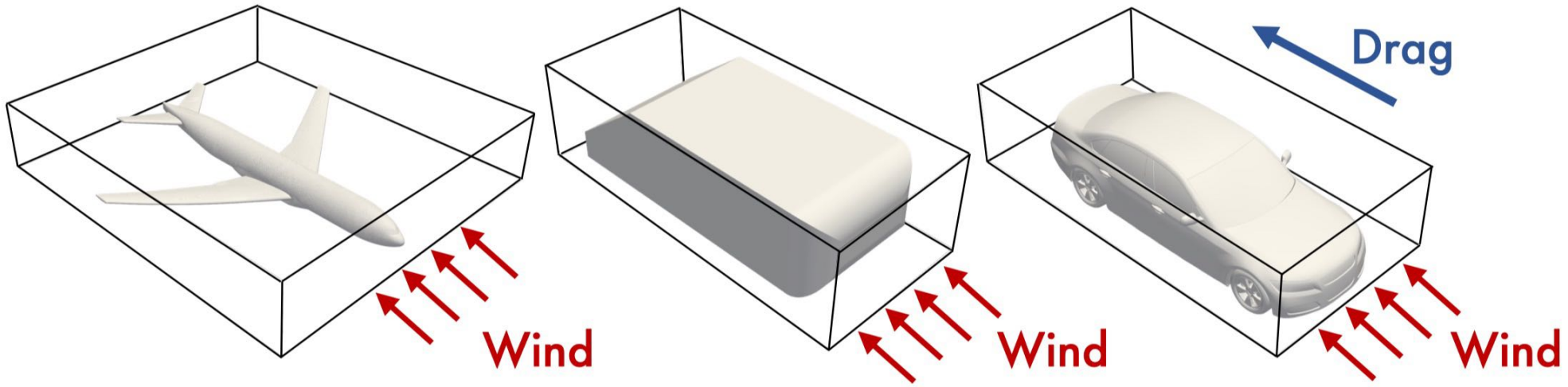
5x larger than vanilla Transolver, 2x larger than Transolver++

Efficiency Analysis (Inference Latency)





Experiments



(a) NASA-CRM

400K cells per sample

4 GB

(b) AhmedML

20M cells per sample

8 TB

(c) DrivAerML

160M cells per sample

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.

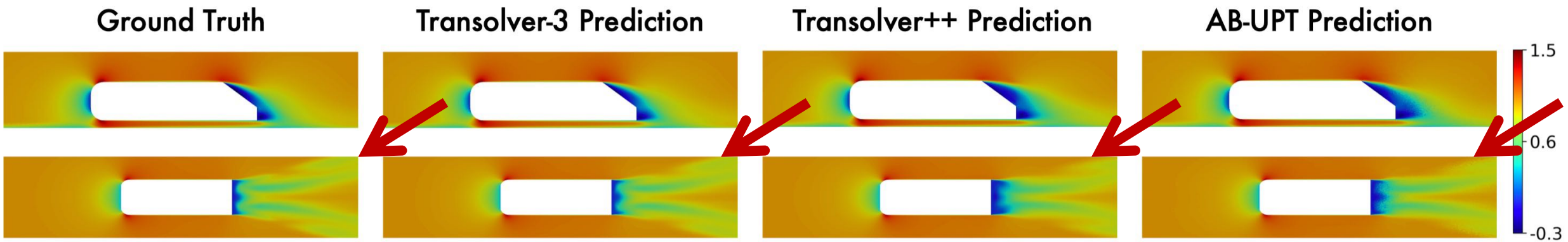
Models	NASA-CRM		AhmedML				DrivAerML			
	$\boldsymbol{p}_s$	$\boldsymbol{C}_f$	$\boldsymbol{p}_s$	$\boldsymbol{u}$	$\boldsymbol{\tau}$	$\boldsymbol{p}_v$	$\boldsymbol{p}_s$	$\boldsymbol{u}$	$\boldsymbol{\tau}$	$\boldsymbol{p}_v$
Graph U-Net*	15.85	15.61	6.46	4.15	7.29	5.18	16.13	17.98	27.84	20.51
GINO*	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.

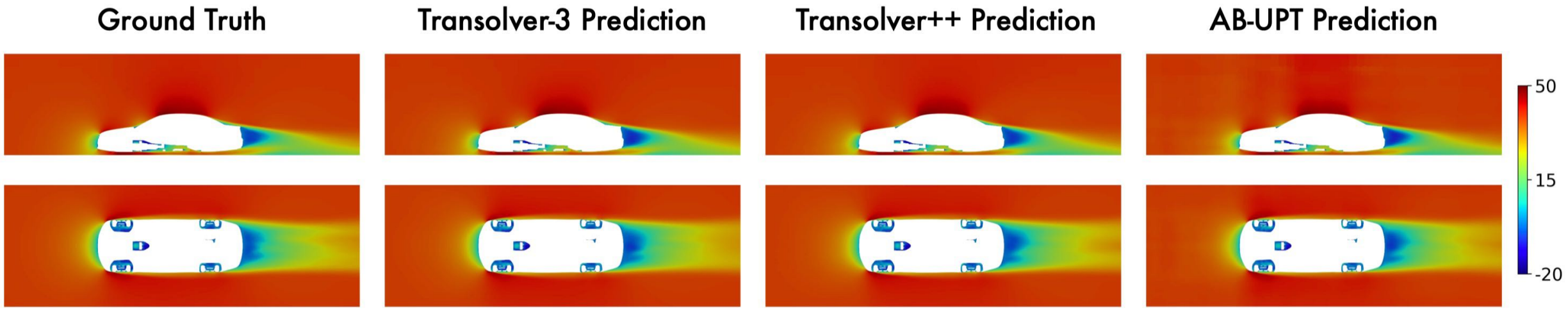
Transolver still achieves the best performance.

Showcase study

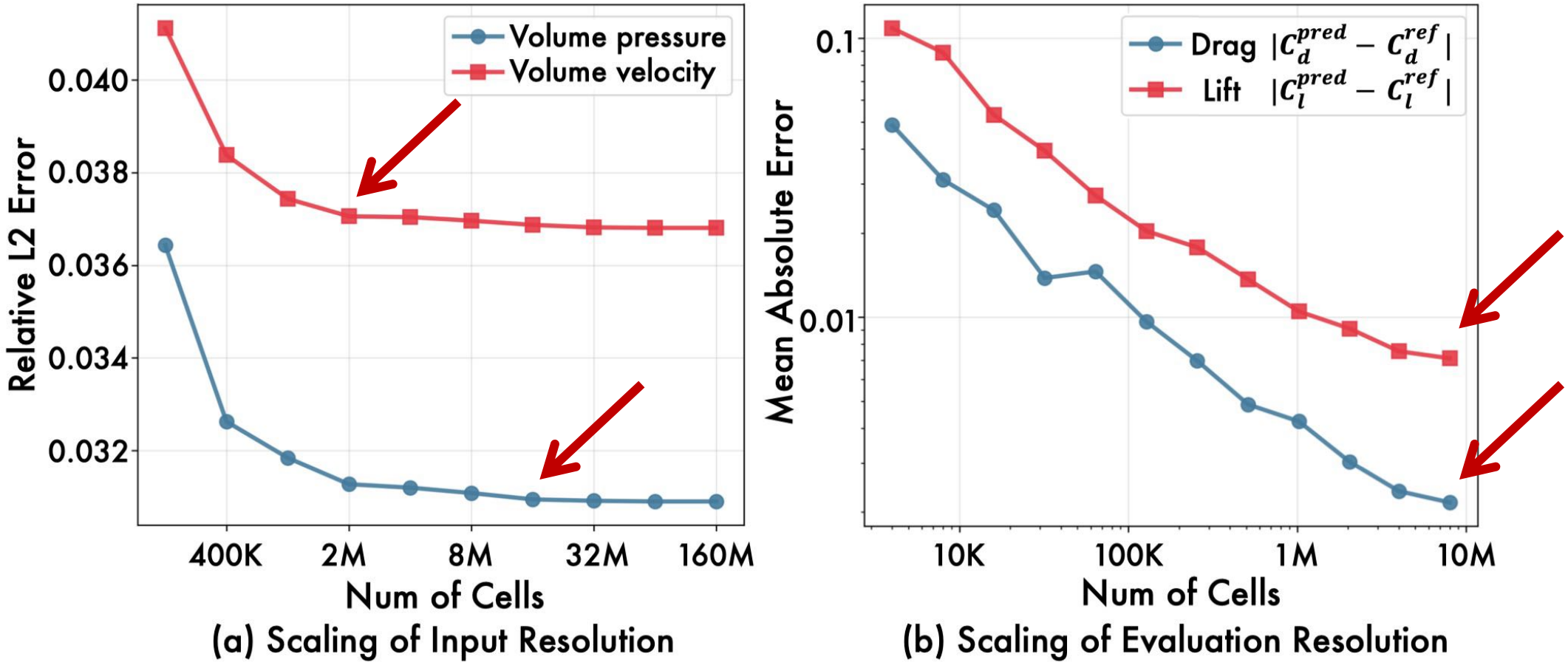
(1) AhmedML Benchmark



(2) DrivAerML Benchmark



Why Geometry Scaling



Scaling Path for Neural Simulators

Scalable Backbone – – – – – – General Geometry — Transolver

Geometry Scaling – – – – – – – Industrial-scale Mesh — Transolver++, Trasolver-3

Physics Scaling – – – – – – – – – Large-scale Pre-training — GeoPT

ICML | 2026
The Forty-first International Conference on Machine Learning

GeoPT: Scaling Physics Simulation via Lifted Geometric Pre-Training

Haixu Wu [*1] **Minghao Guo** [*1] **Zongyi Li** [1] **Zhiyang Dou** [1] **Mingsheng Long** [2] **Kaiming He** [1] **Wojciech Matusik** [1]


Haixu Wu*


Minghao Guo*


Zongyi Li


Zhiyang Dou


Mingsheng Long

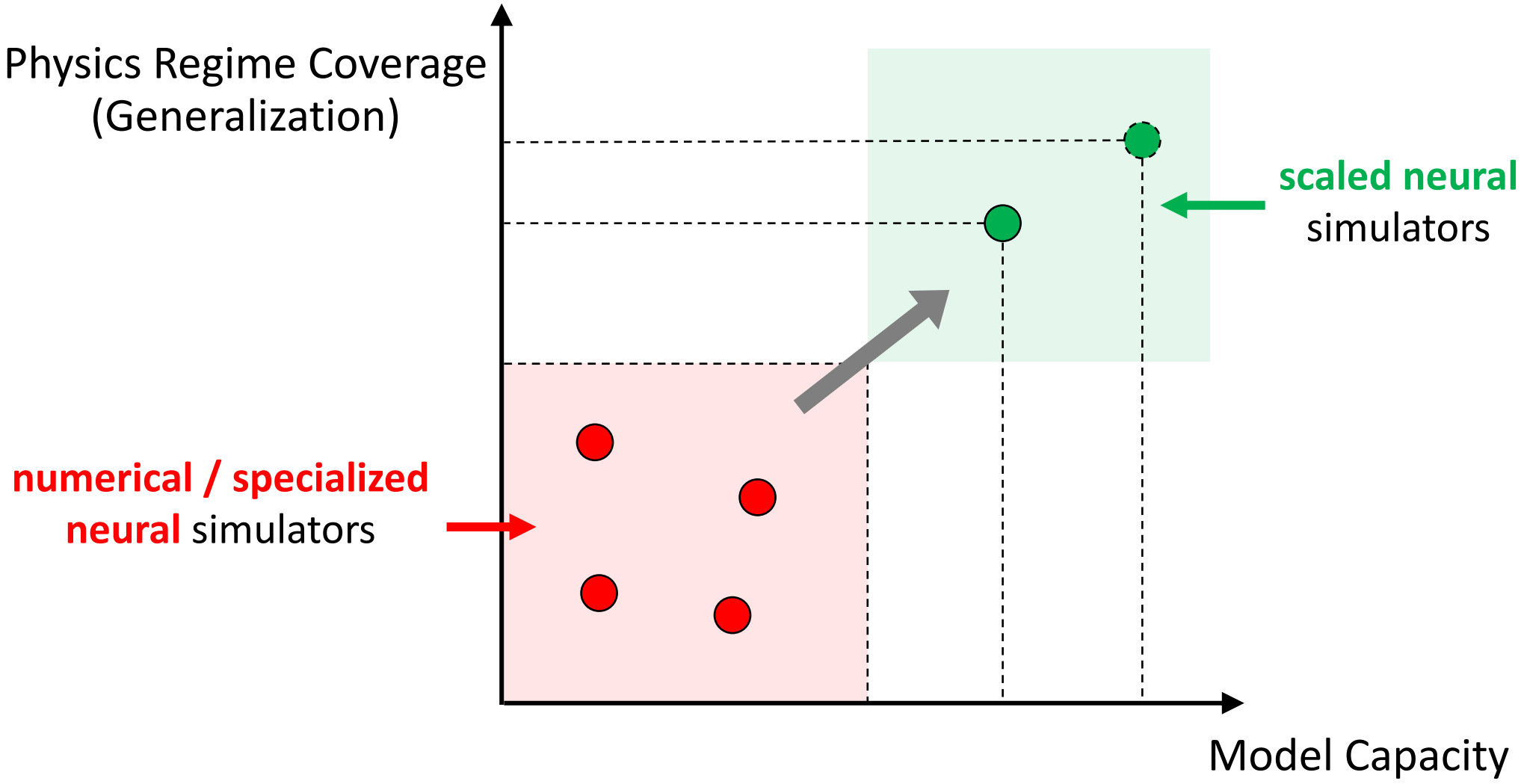

Kaiming He


Wojciech Matusik

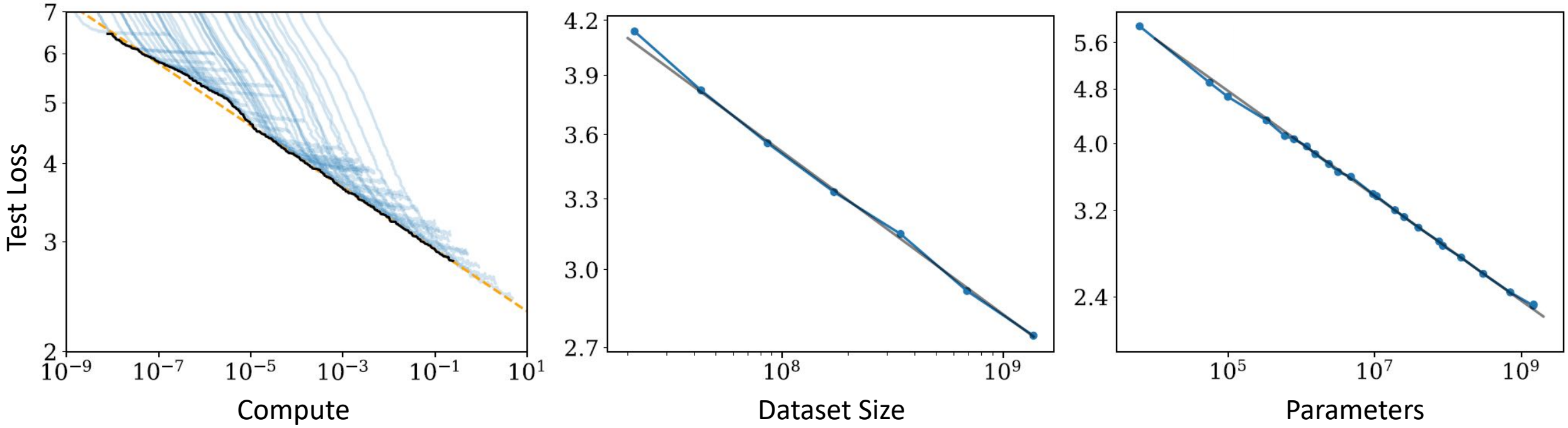
Code Link: https://github.com/Physics-Scaling/GeoPT

🏆 **Best Paper Award** at ICLR 2026 Workshop on Foundation Model for Science

Scaling up Neural Simulators



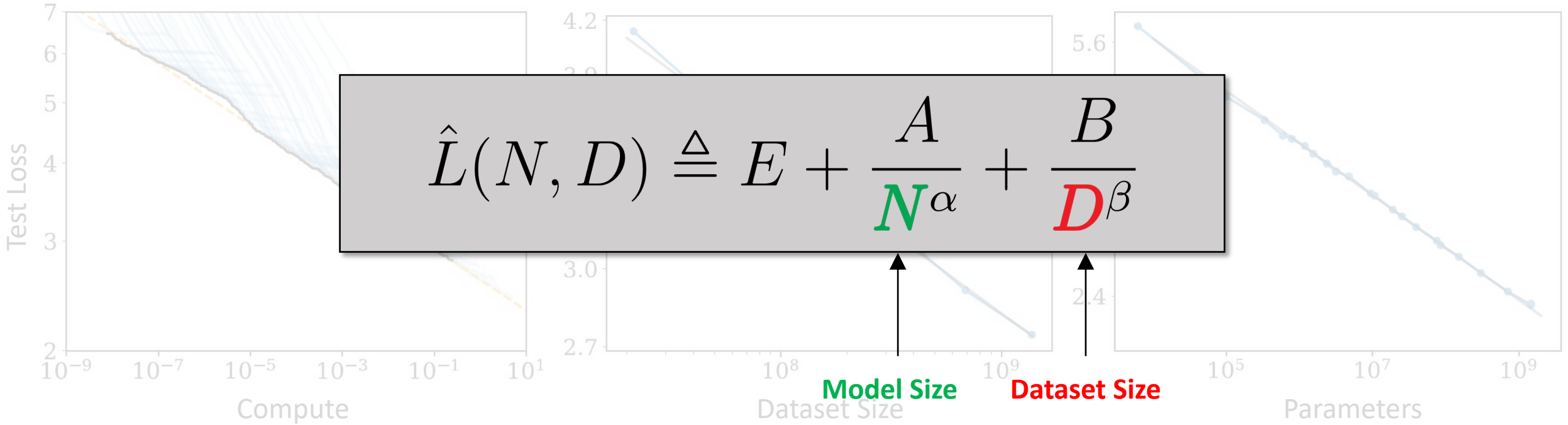
Scaling Law in Large Language Models



[Kaplan et al., Scaling Laws for Neural Language Models, arXiv 2020]

[Hoffmann et al., Training Compute-Optimal Large Language Models, arXiv 2022]

Scaling Law in Large Language Models

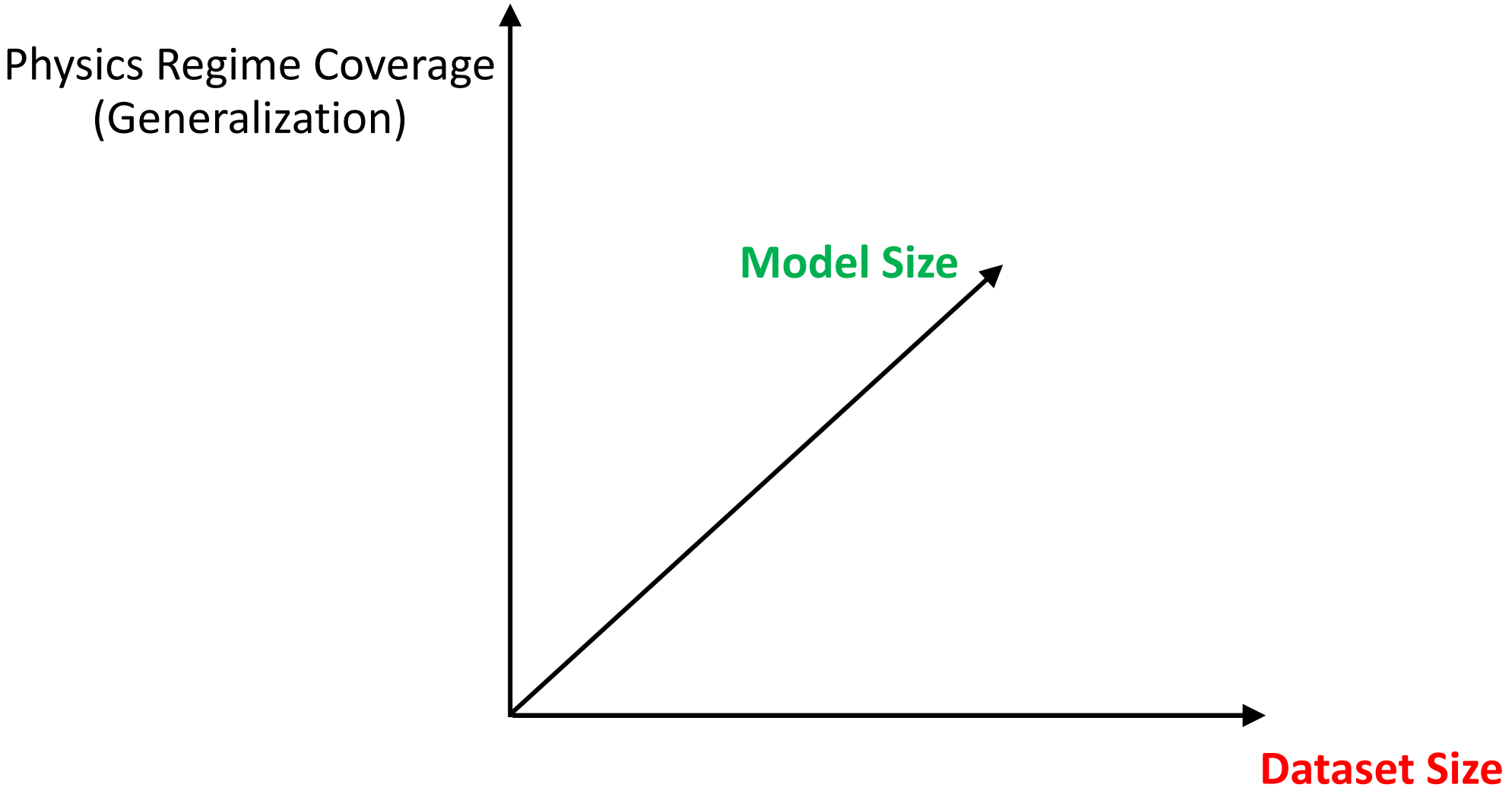


$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

[Kaplan et al., Scaling Laws for Neural Language Models, arXiv 2020]
[Hoffmann et al., Training Compute-Optimal Large Language Models, arXiv 2022]

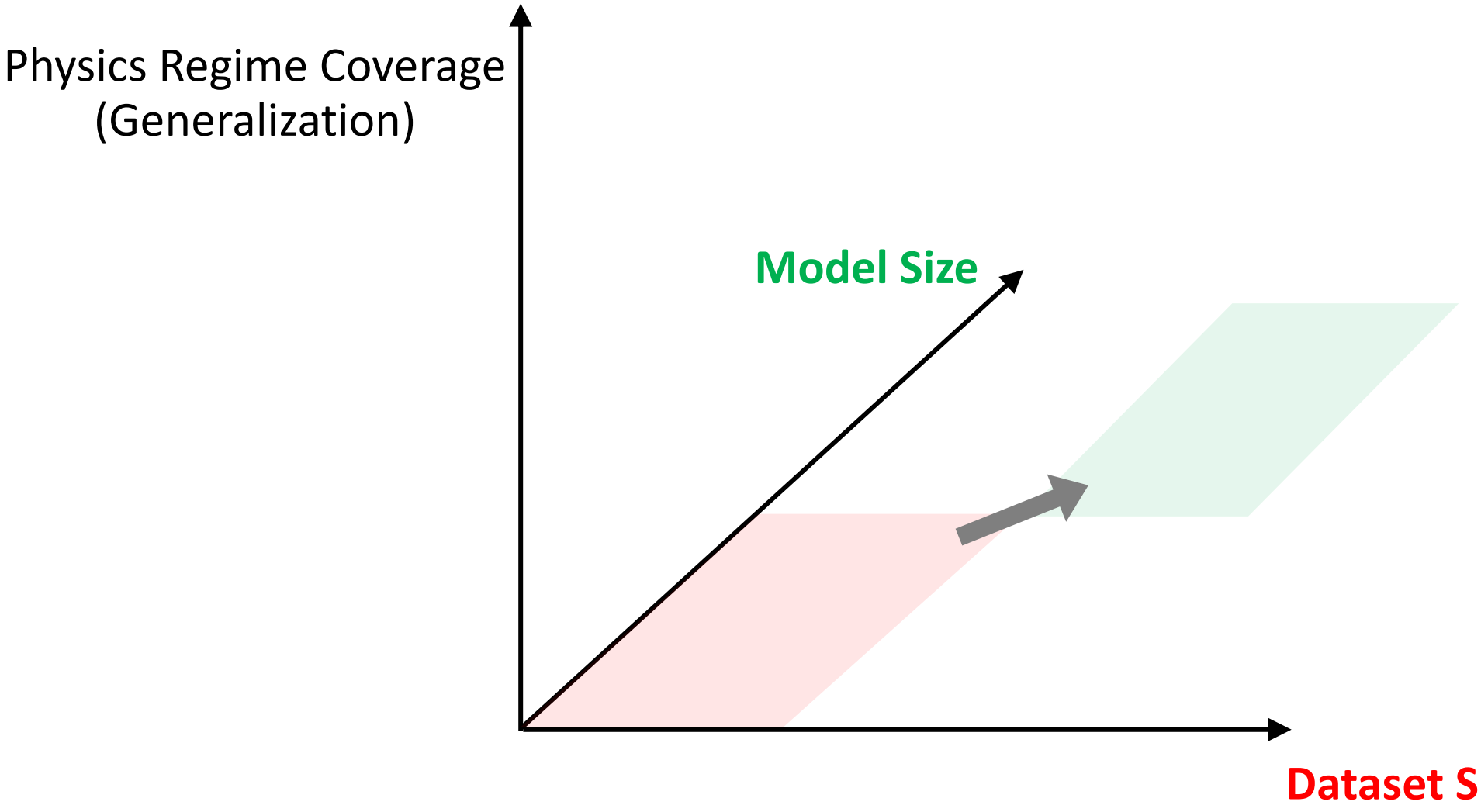
Scaling Law in Neural Simulators

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$



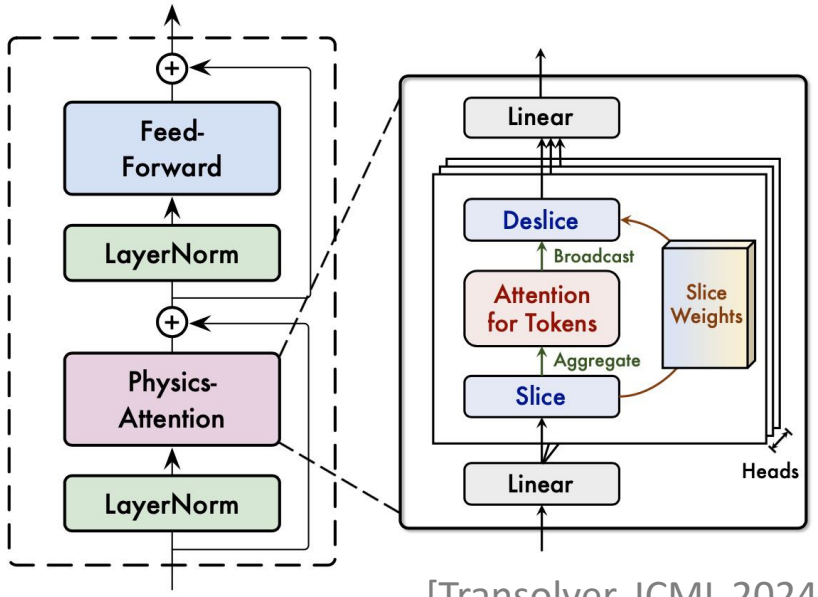
Scaling Law in Neural Simulators

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$



Scaling Law in Neural Simulators

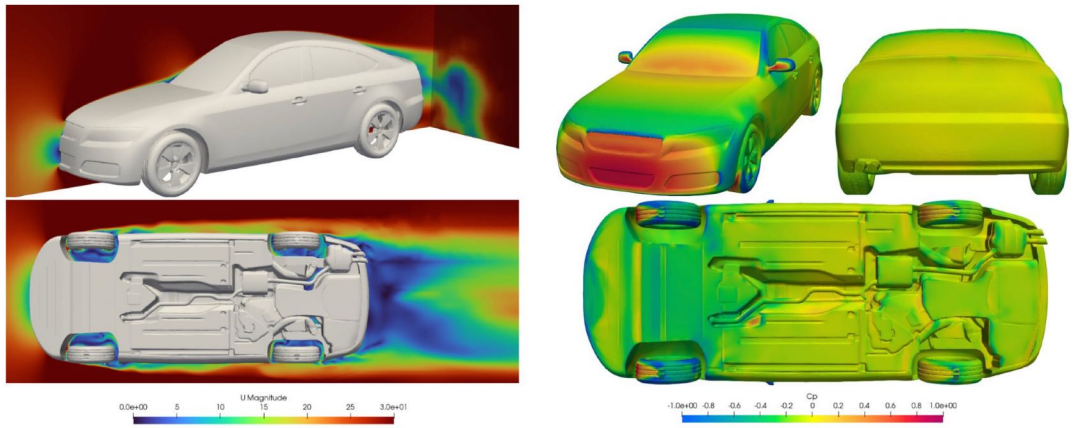
Model Size



[Transolver, ICML 2024]

Increase **depth + width**

Dataset Size



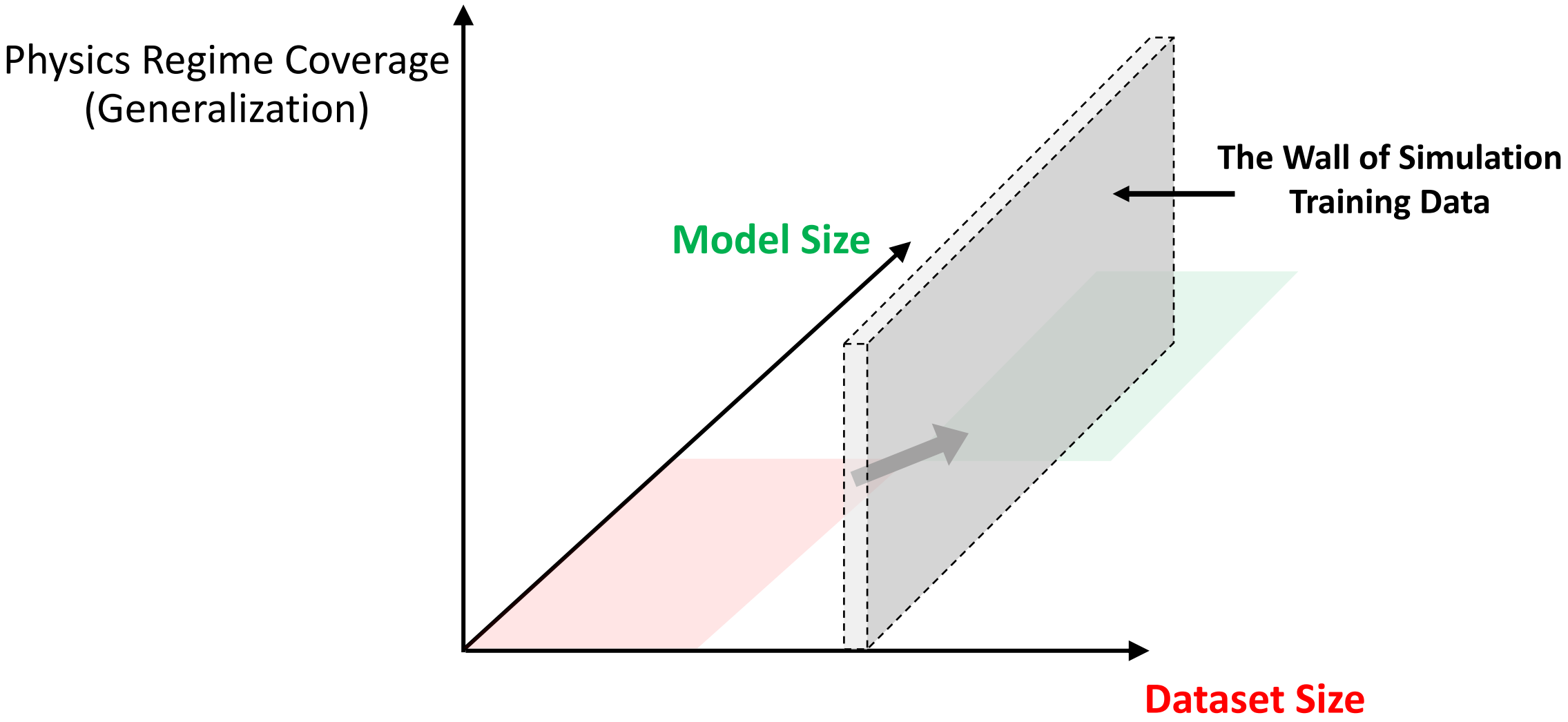
[Source: DrivAernet++, DrivAerML, NASA-CRM …]

A typical simulation data took around

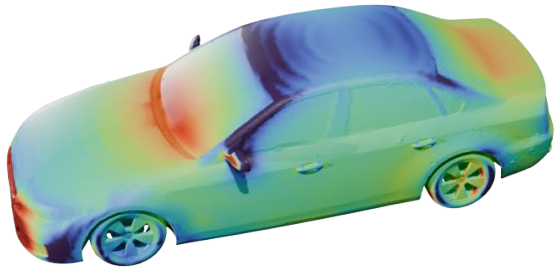
6.1×10^4 CPU-hours

Scaling Law in Neural Simulators

$$\hat{L}(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}$$

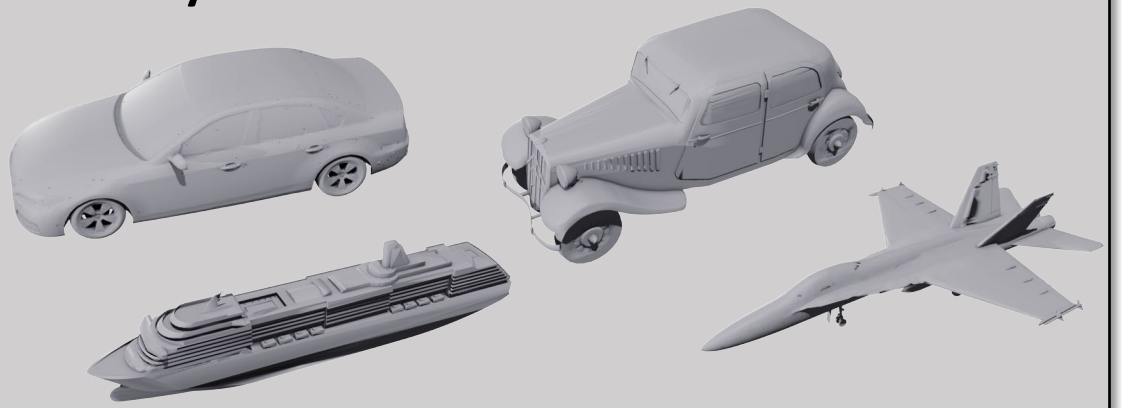


Data of Physics Simulation



Physics Data

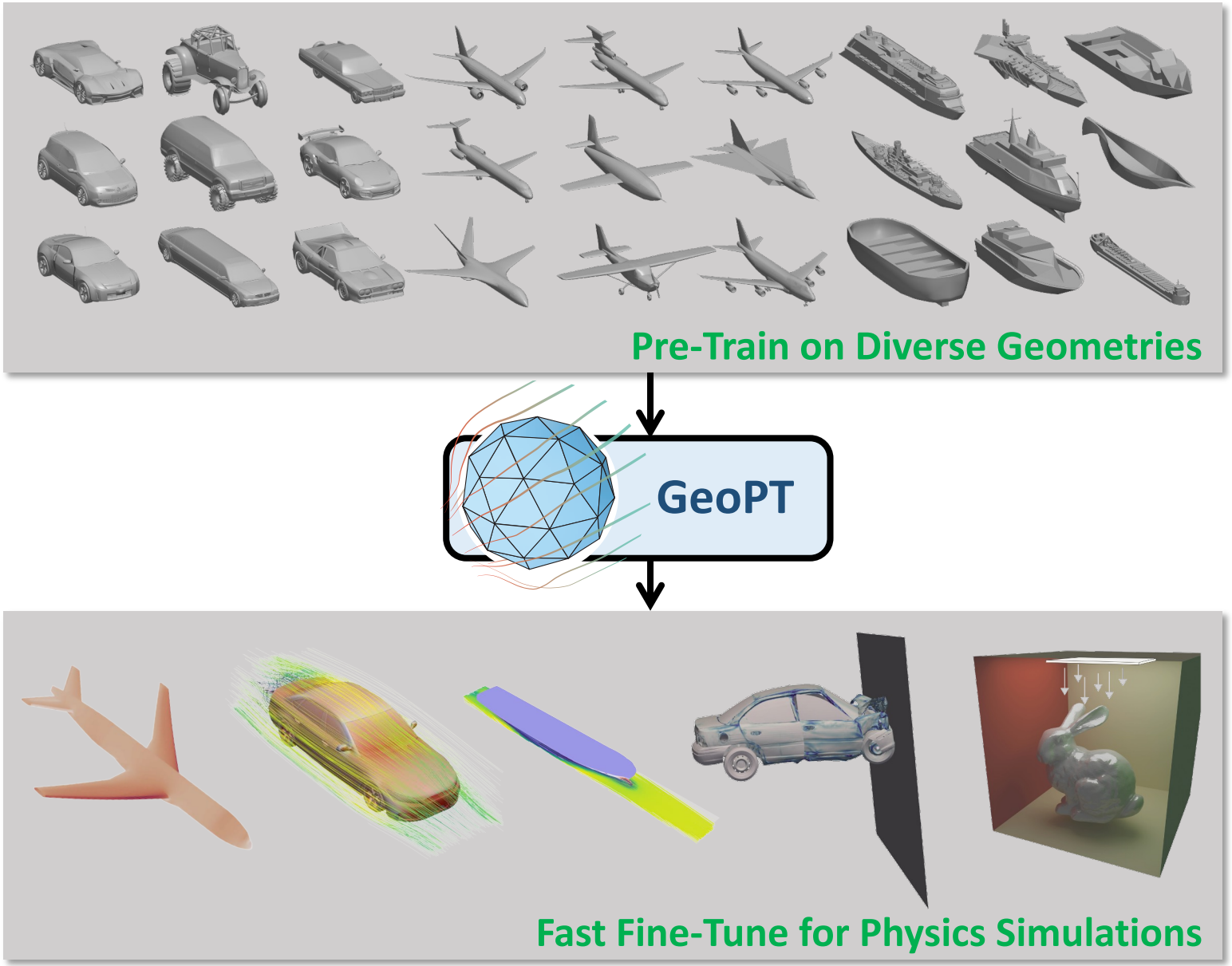
Geometry **Abundant at Scale**



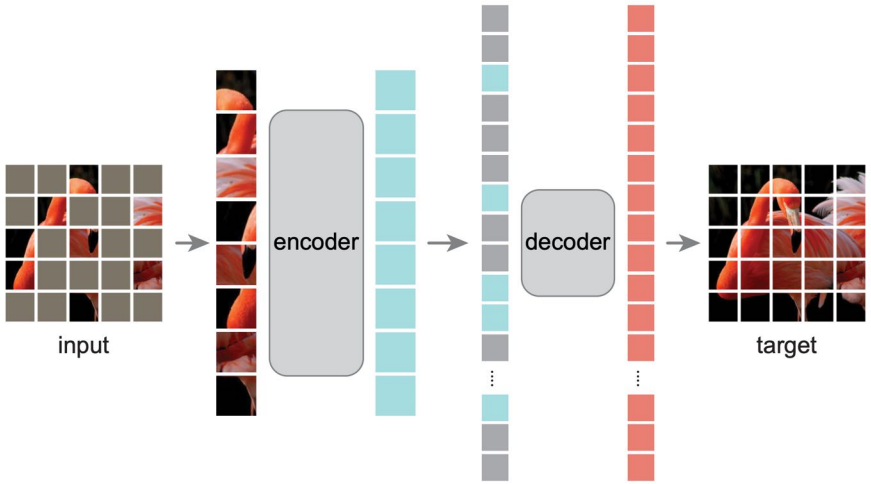
[OnShape, ShapeNet, Objaverse…]

Dynamics **Computational Expensive**

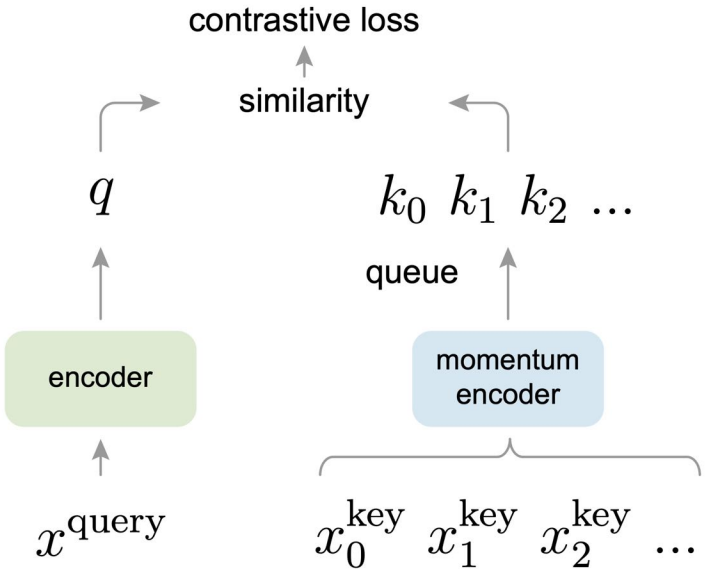
Geometric Pre-Training



Self-supervised Pre-training Paradigm



MAE [He et al., CVPR 2022]

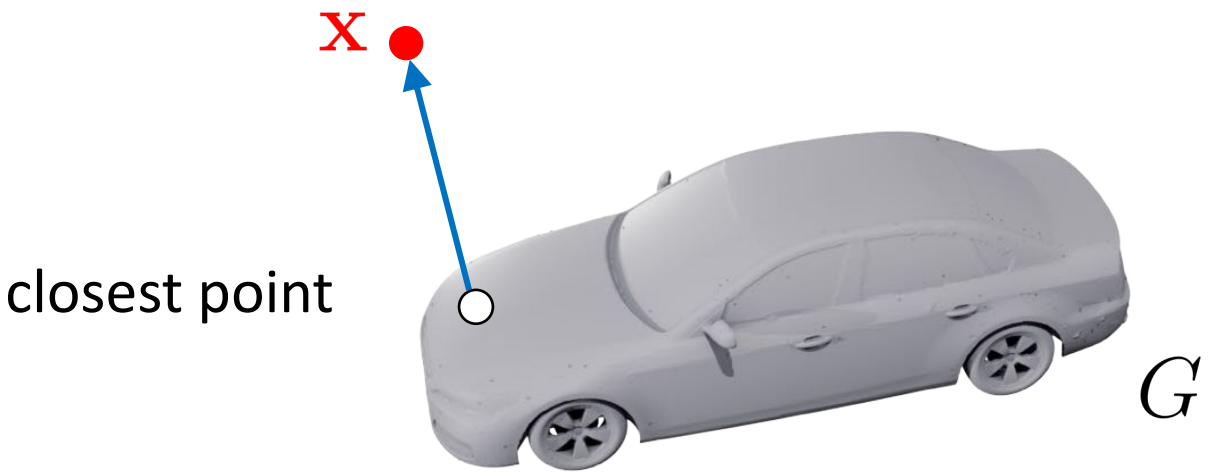


MoCo [He et al., CVPR 2020]

Input/View Construction

Pretext Learning Objective

Geometry Only Pre-training



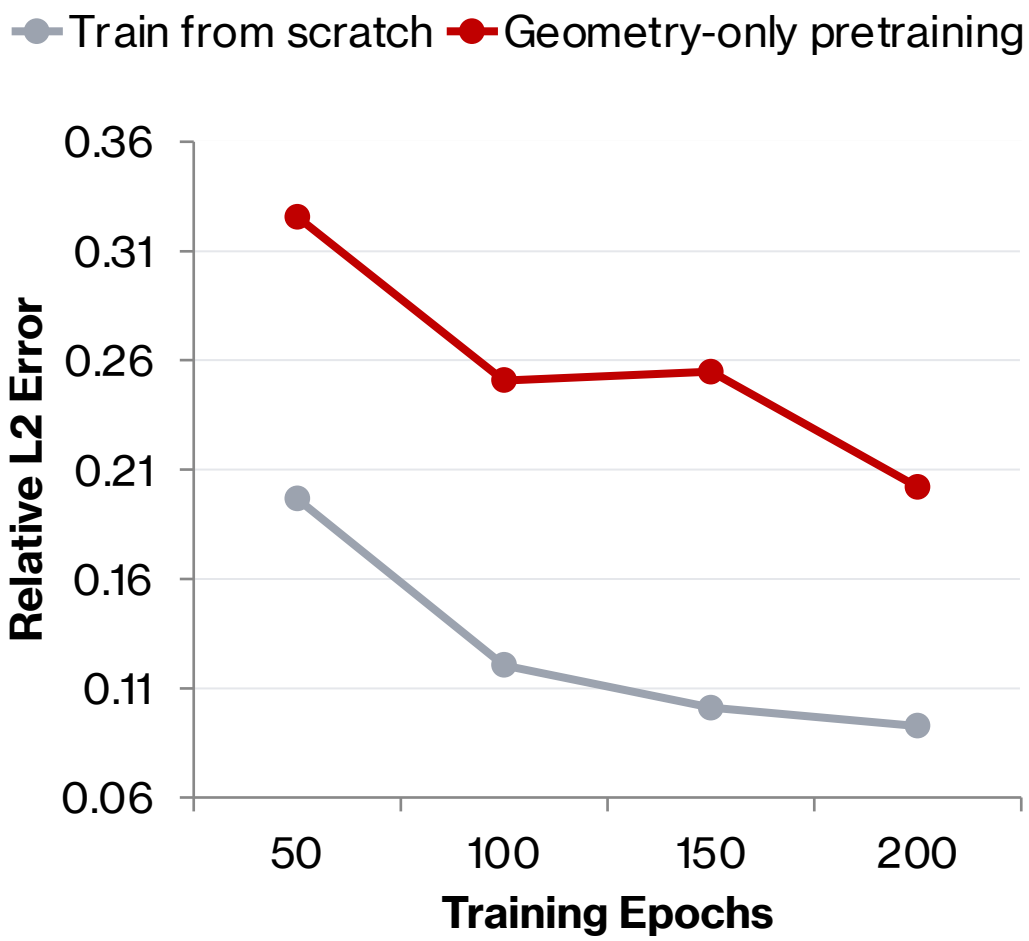
$$\mathcal{L}^{\text{pre}}_{\text{native}} = \mathbb{E}_{\mathbf{x},G}\left[\|\mathcal{F}_{\hat{\theta}}(\mathbf{x};G) - h_G(\mathbf{x})\|_2^2\right]$$

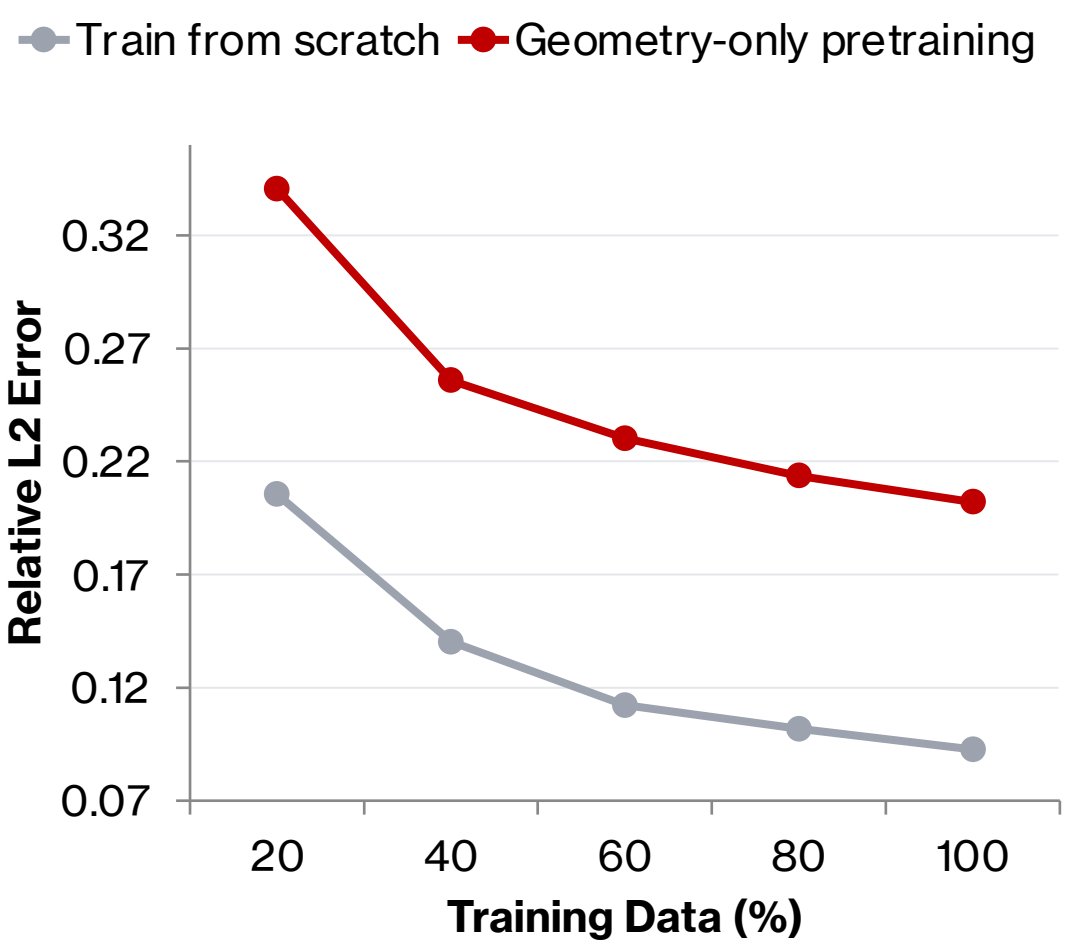
vector distance function

Input/View Construction: **Geometry only**

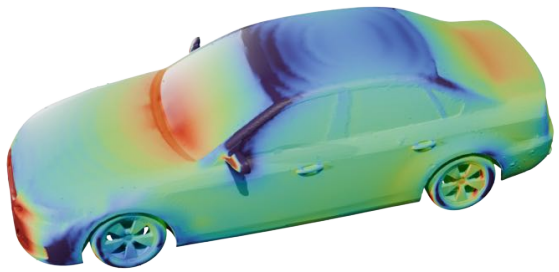
Pretext Learning Objective: **Geometric feature prediction**

Geometry Only Pre-training Fails

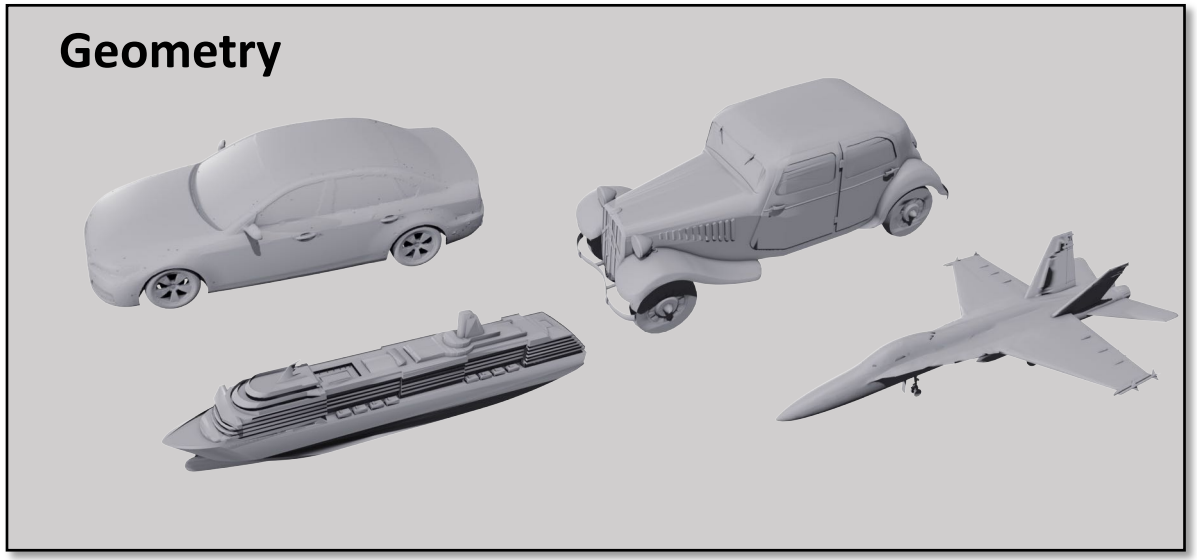




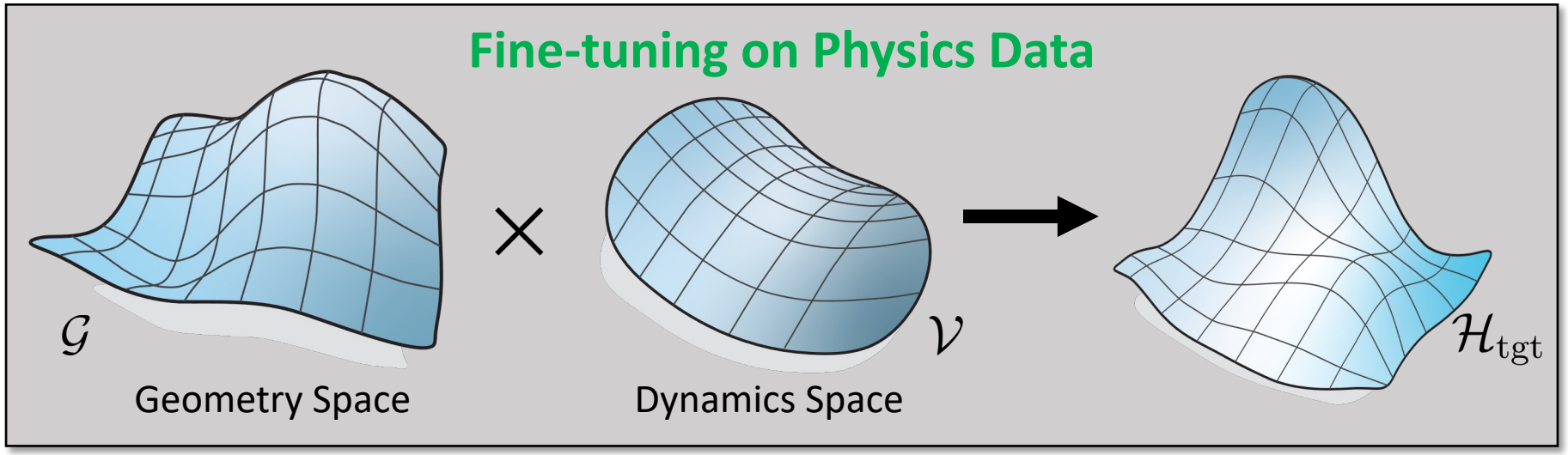
Why Geometry Only Pre-training Fails

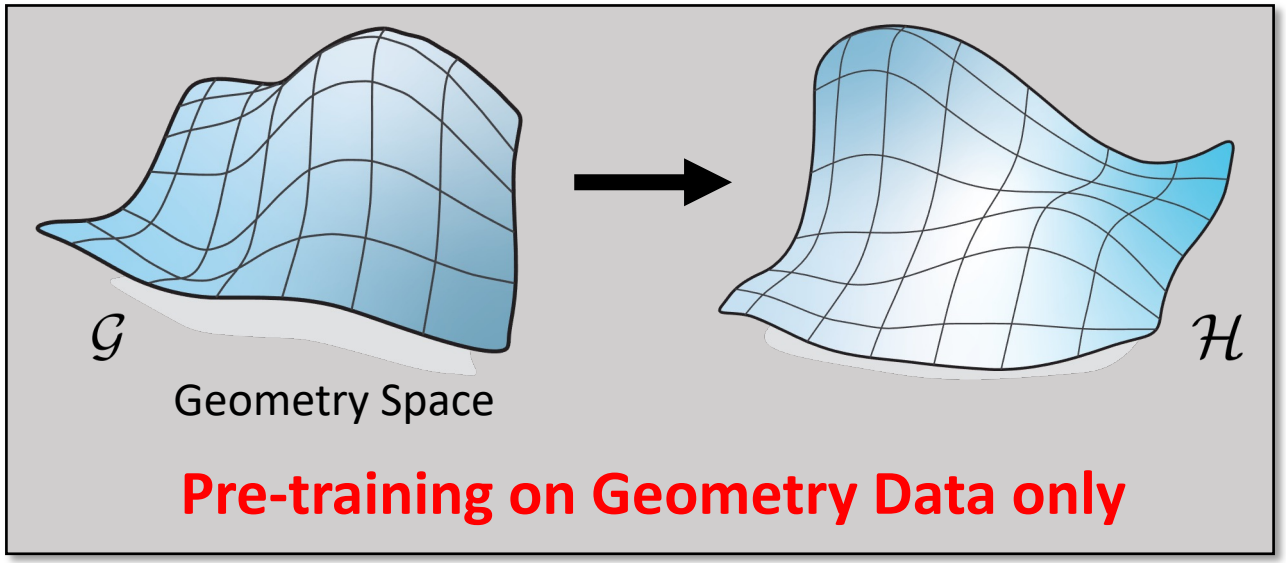


Physics Data

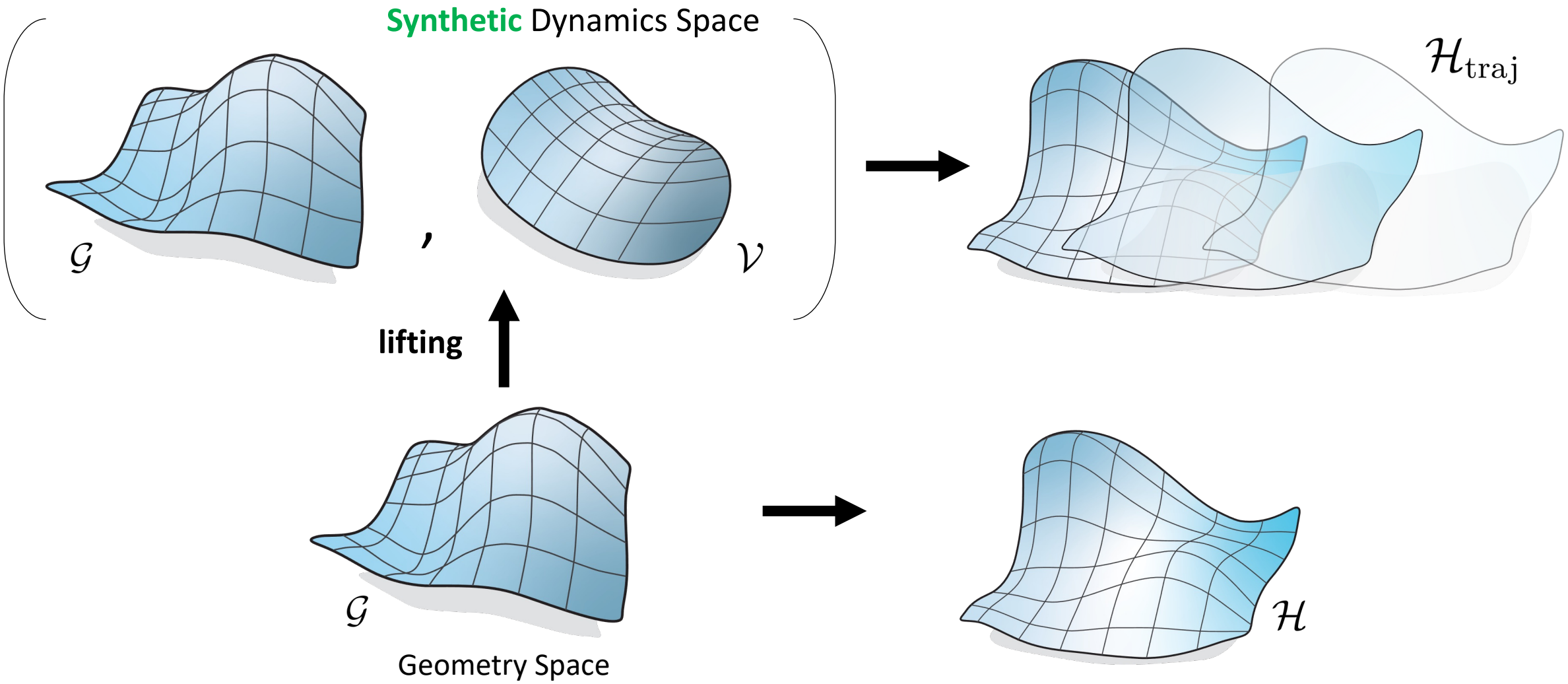


Pre-training and Fine-tuning live in Different Spaces

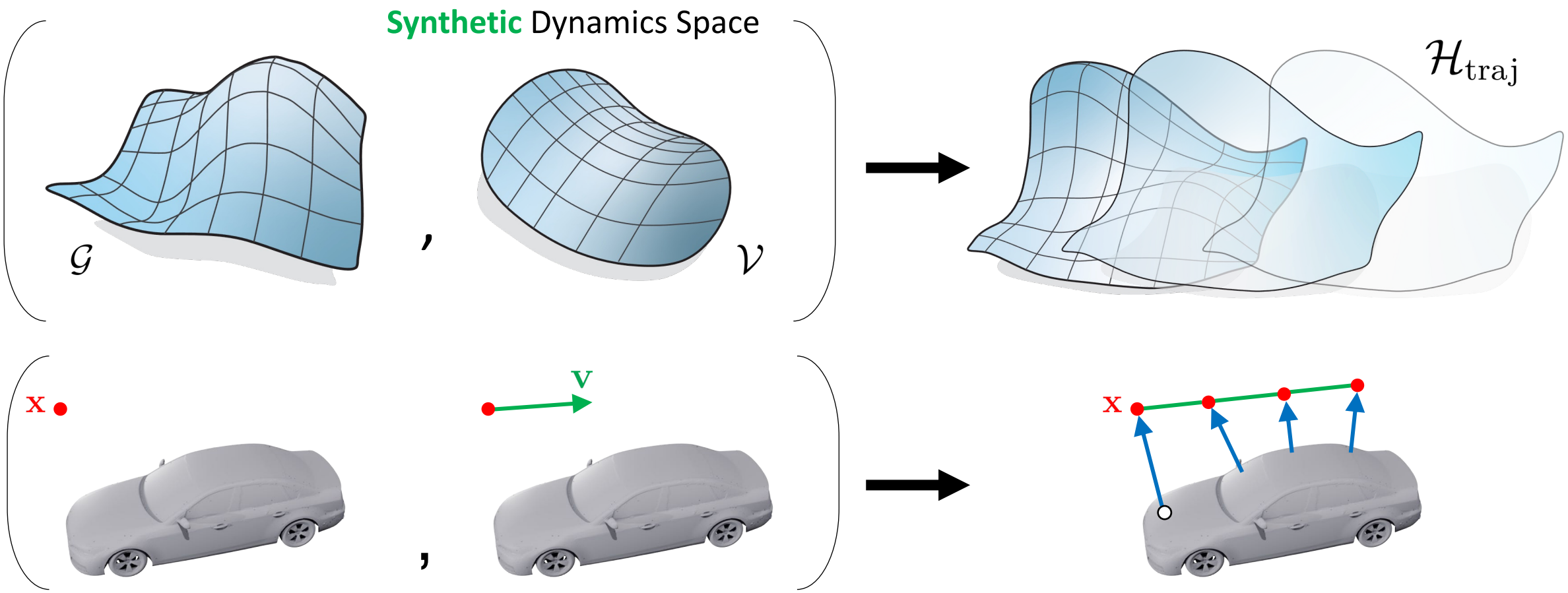




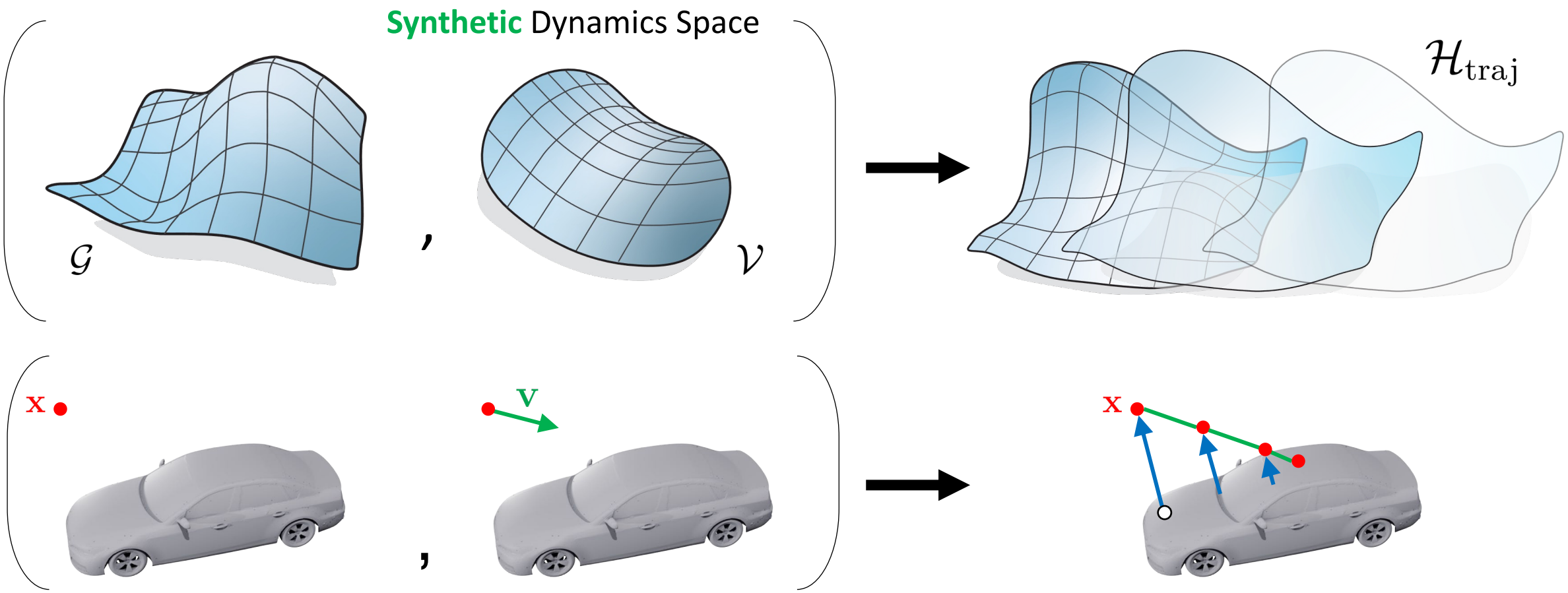
Our Solution: Lifted Geometric Pre-Training



Our Solution: Lifted Geometric Pre-Training

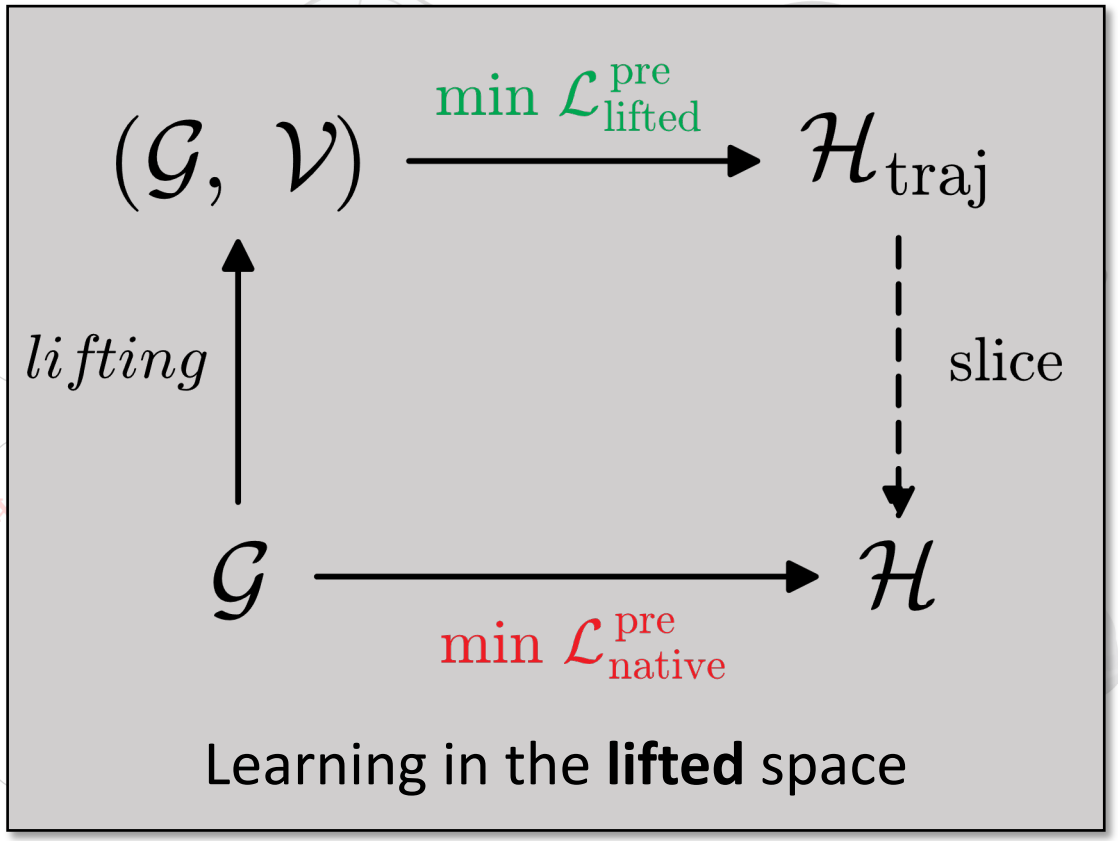


Our Solution: Lifted Geometric Pre-Training



$$\frac{\mathrm{d}\mathbf{x}_t}{\mathrm{d}t} = \mathbf{v} \cdot \mathbb{1}_G(\mathbf{x}_t), \quad \mathbf{x}_0 = \mathbf{x}, \quad \mathbf{v} \sim \mathrm{Unif}(\mathbb{B}^C), \qquad \boldsymbol{h}_G(\mathbf{x}_{0:\tau}) = \{\boldsymbol{h}_G(\mathbf{x}_t)\}_{t=0}^{\tau} \in \mathcal{H}_{\text{traj}}$$

Our Solution: Lifted Geometric Pre-Training



Synthetic Dynamics
(Transport Equation):

$$\partial_t f + v \cdot \nabla_x f = 0$$

The only inductive bias:

mass conservation

GeoPT: Pre-Training and Fine-Tuning

Self-supervised Pre-training

$$\mathcal{L}^{\text{pre}}_{\text{lifted}} = \mathbb{E}_{\mathbf{x},\,G,\,V}\left[\left\|\mathcal{F}_{\widehat{\theta}}(\mathbf{x};\,G,\,V) - \boldsymbol{h}_G(\mathbf{x}_{0:\tau})\right\|_2^2\right]$$

Synthetic Dynamics

Real Dynamics (inflow speed, angle of attack,…)

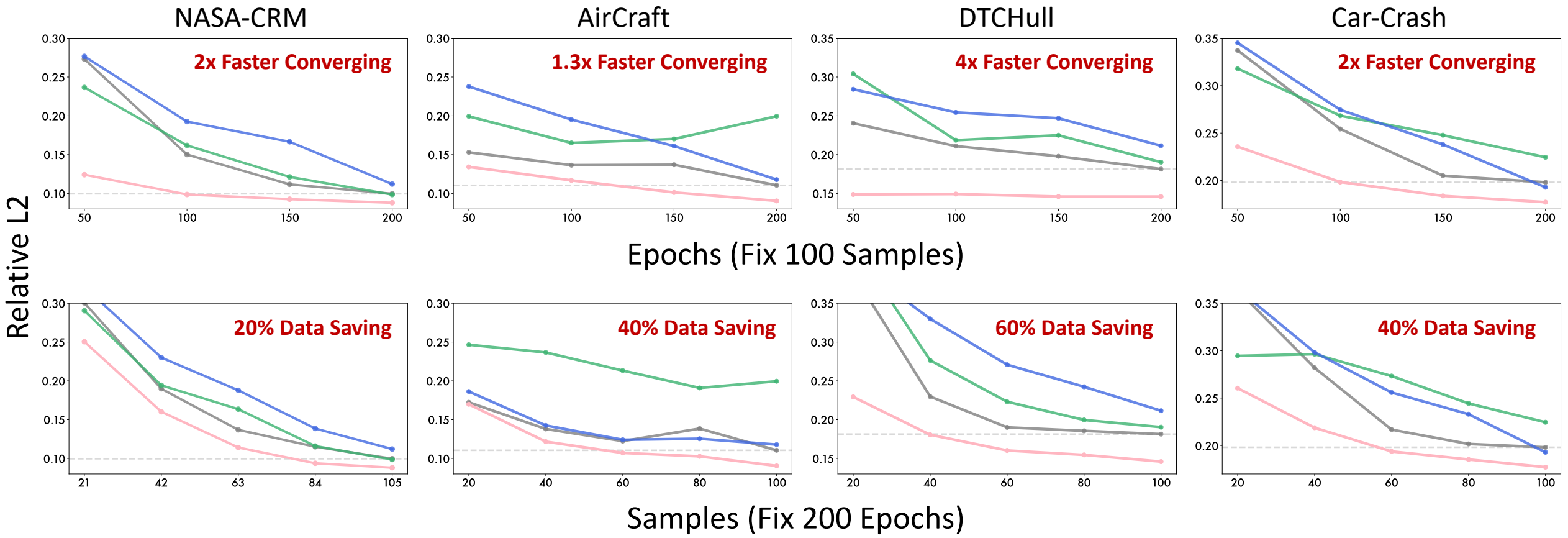
$$\mathcal{L}^{\text{fine}} = \mathbb{E}_{\mathbf{x},\,G,\,V_S}\left[\left\|\mathcal{F}_{\theta}(\mathbf{x};\,G,\,V_S) - \boldsymbol{u}(\mathbf{x})\right\|_2^2\right]$$

Fine-tuning on Physics Data

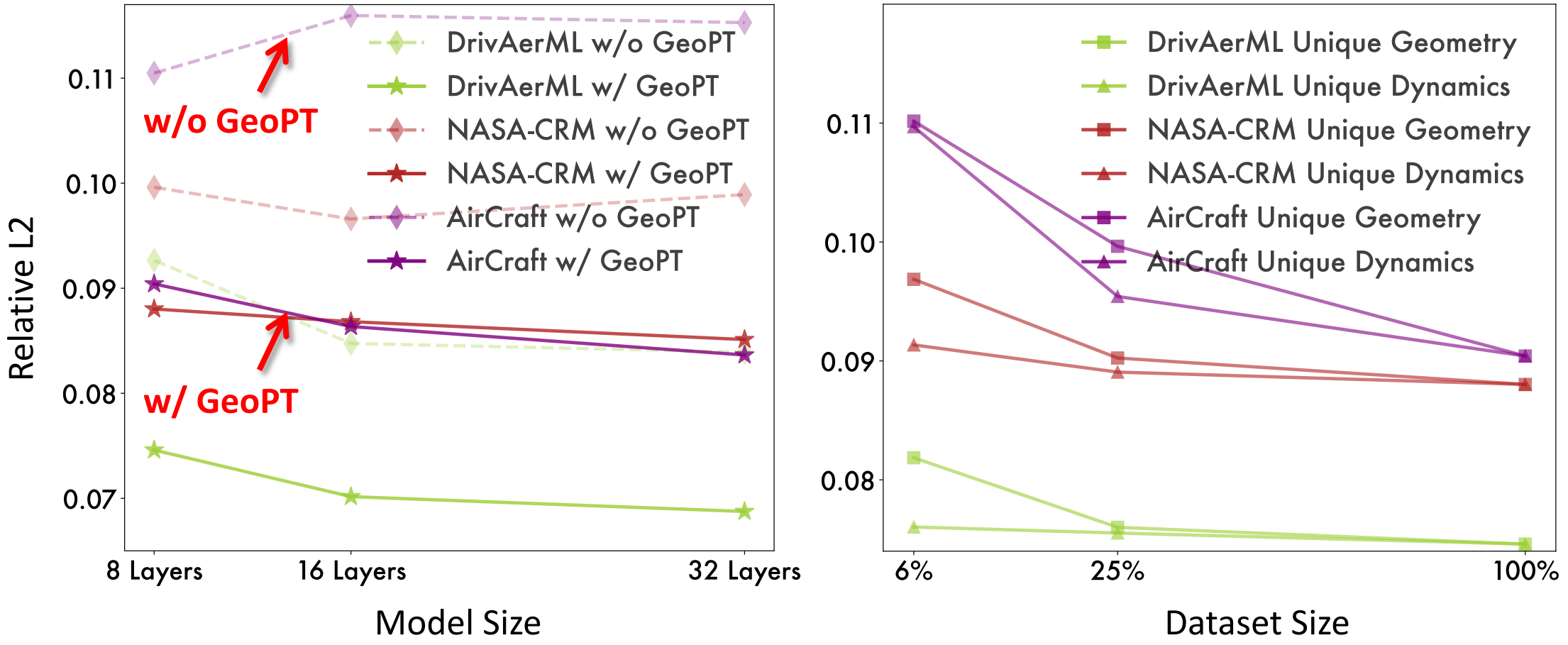
Compute self-supervision by ray-tracing

Generating million-scale data in 3 days (0.2s per sample)

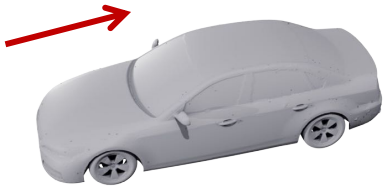
Results: Faster Convergence, Fewer Data



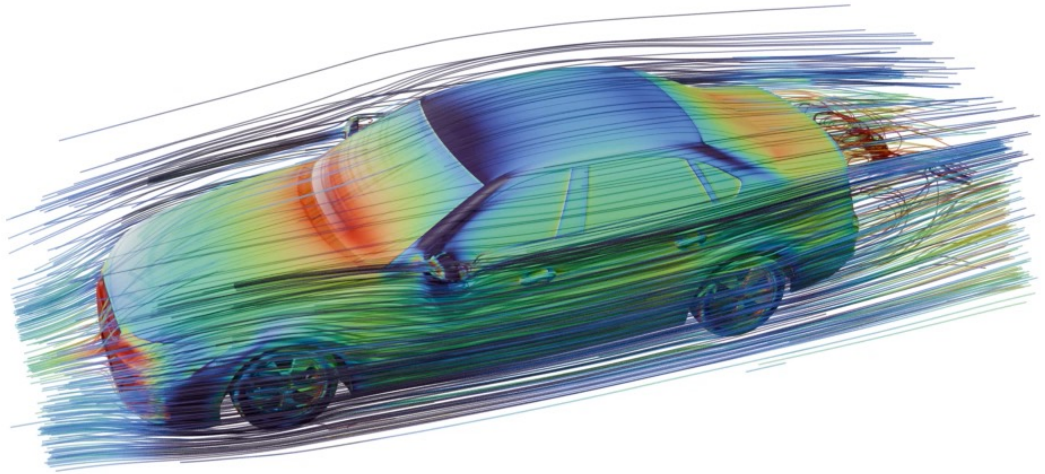
Results: Scalability



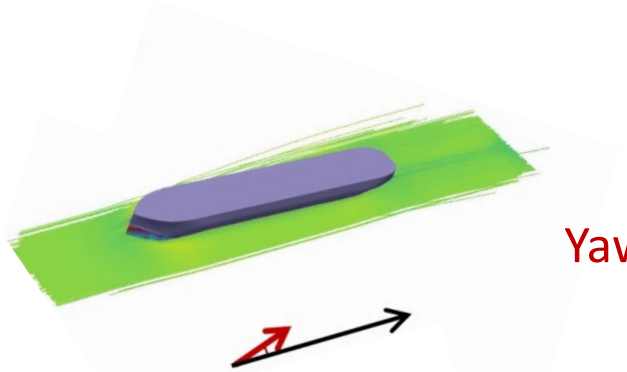
Prediction Visualization



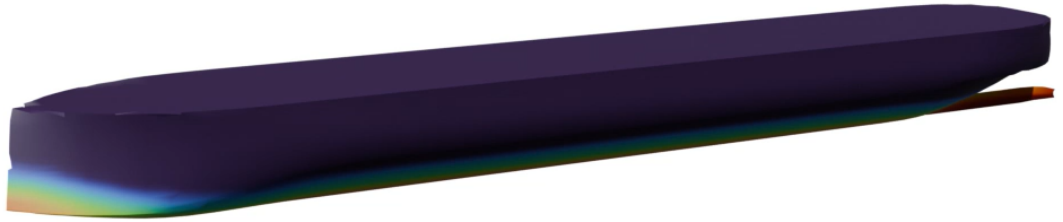
Incoming flow speed
38.89 m/s



GeoPT on DrivAerML

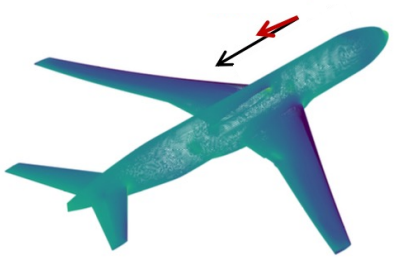


Yaw Angle 11°



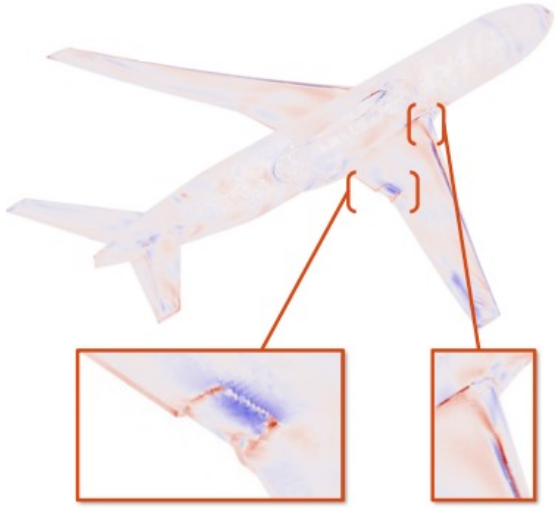
GeoPT on DTCHull

NASA-CRM

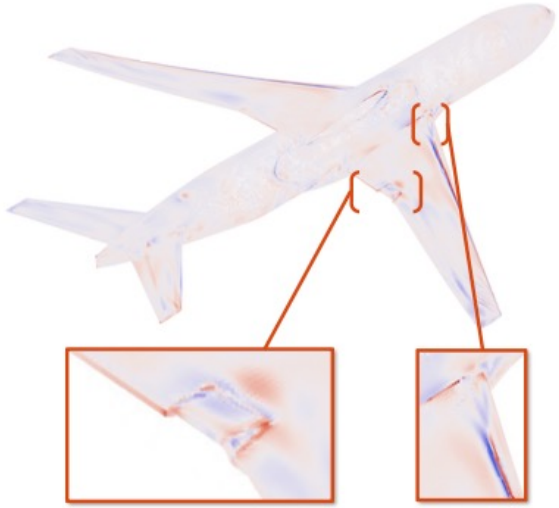


Angle of Attack 26.93°
0.7219 Mach

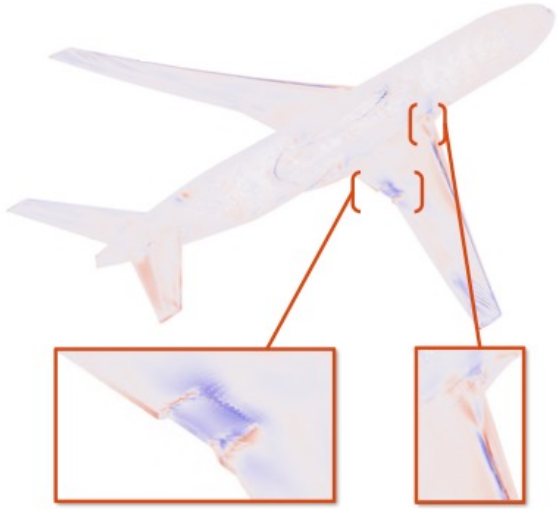
Error Map



Transolver

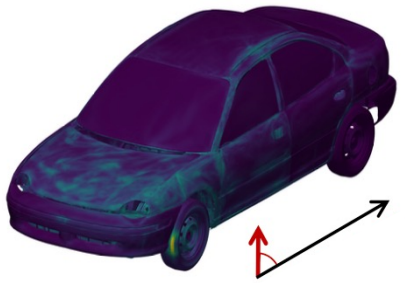


Transolver++



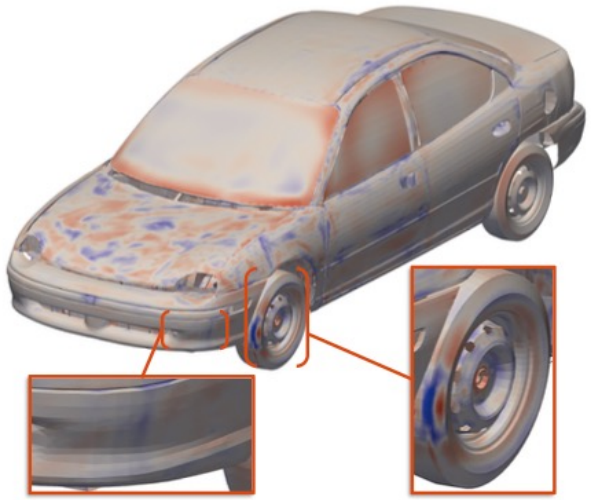
GeoPT

Car-Crash

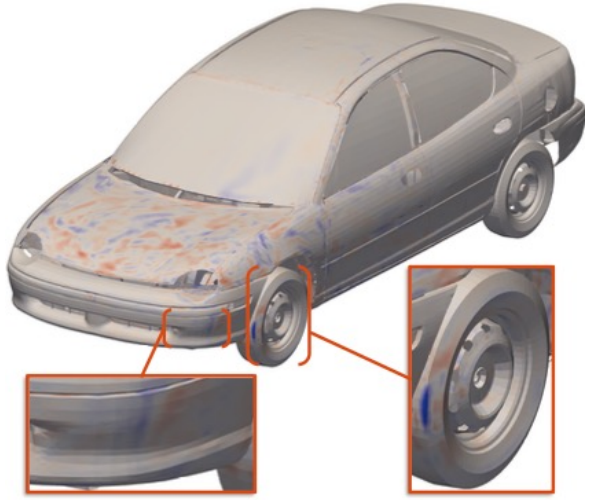


Impact Angle 26.93°

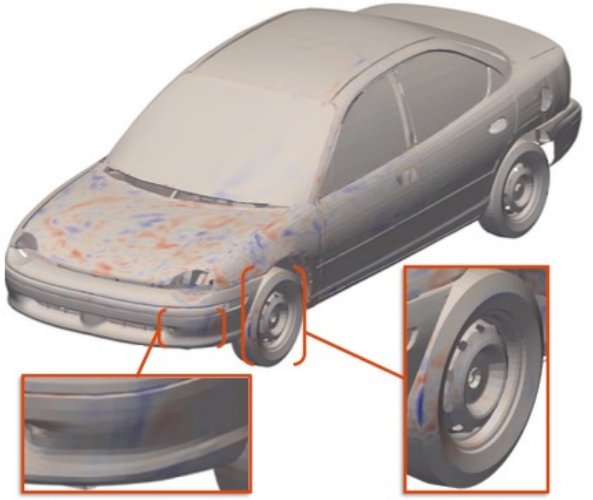
Error Map



UPT

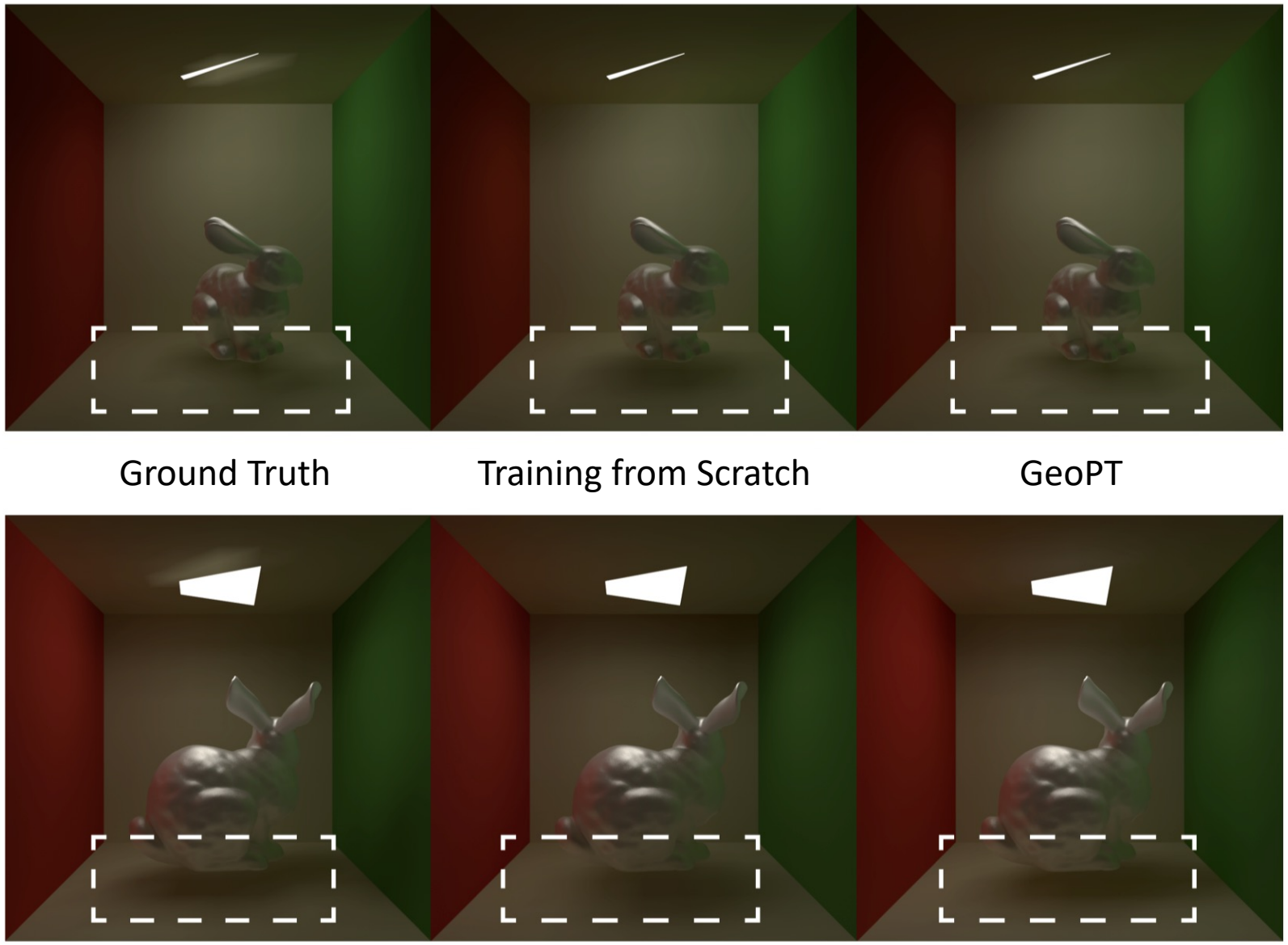


Transolver



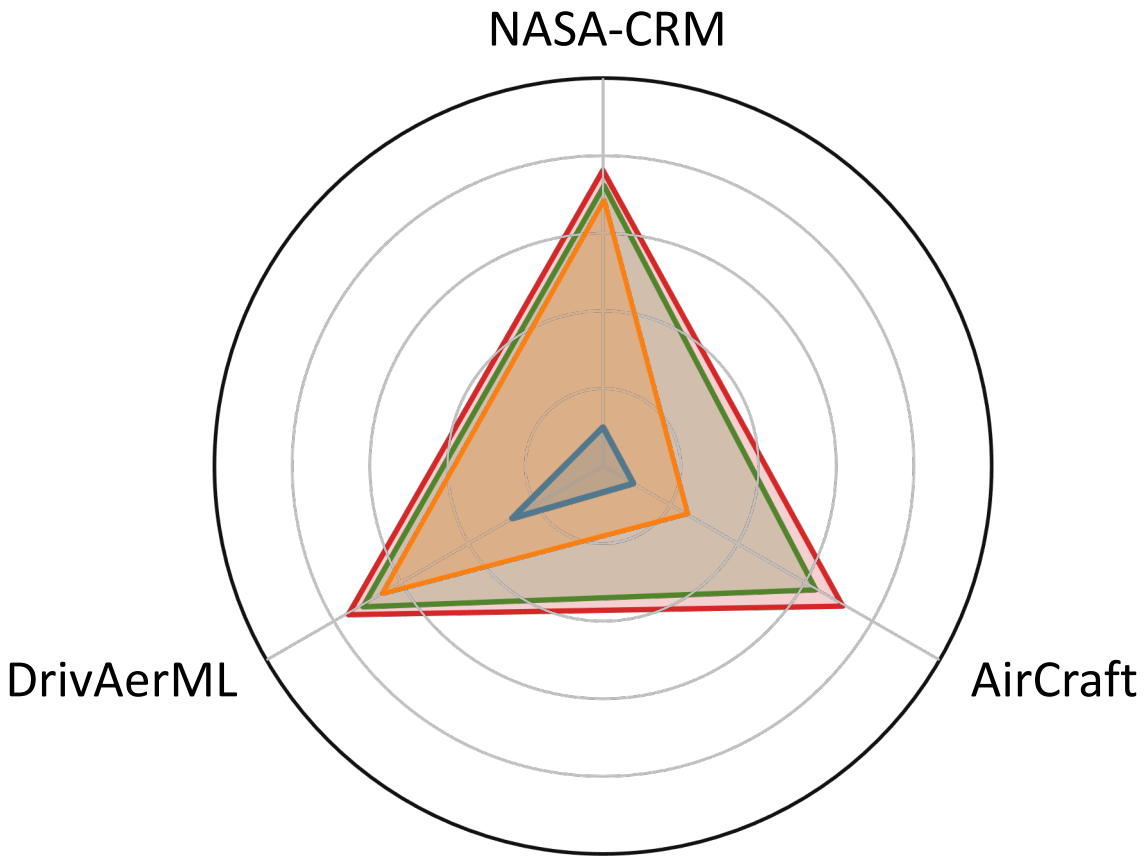
GeoPT

OOD Generalization: Radiosity



Ground Truth | Training from Scratch | GeoPT

Further Scaling Up

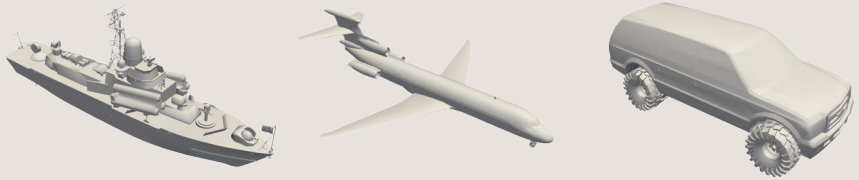


Scaling Performance

Based on GeoPT-Huge model (32 layers)

Train from Scratch

ShapeNet Subset — 3 categories; 10,000 Geos



Full ShapeNet — 55 categories; 50,000 Geos

ShapeNet + HY3D — General categories; 300,000 Geos

ShapeNet [Chang et al., 2015], HY3D-Bench [Hunyuan3D, 2026]

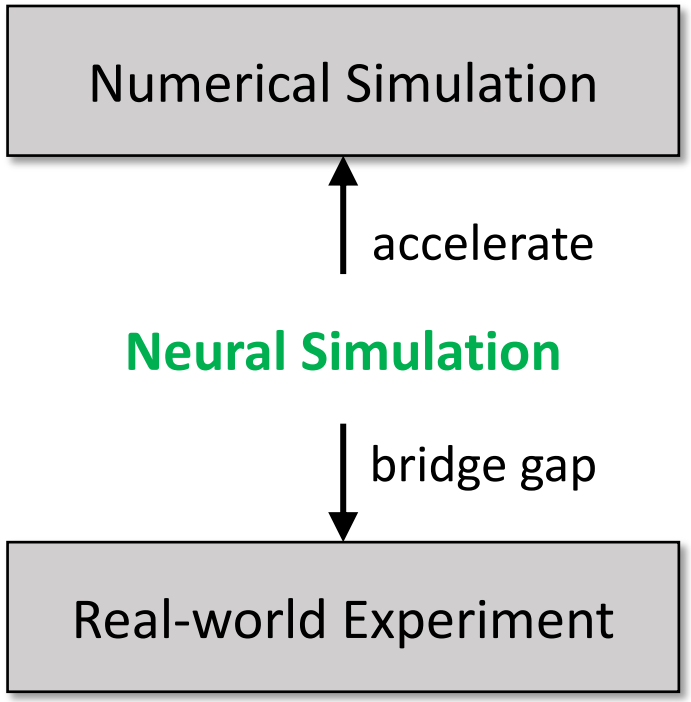
Scaling Path for Neural Simulators

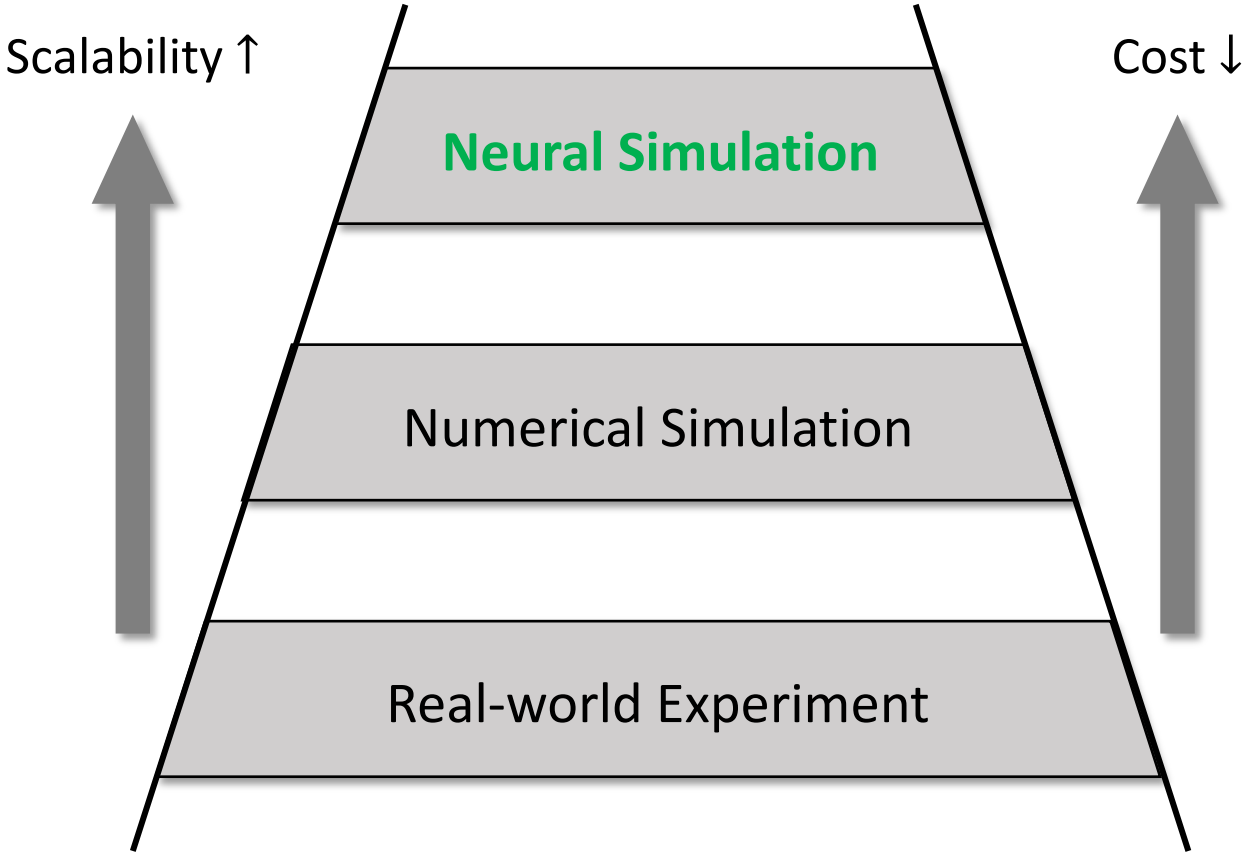
Scalable Backbone – – – – – – **General Geometry** — Transolver

Geometry Scaling – – – – – – – **Industrial-scale Mesh** — Transolver++, Trasolver-3

Physics Scaling – – – – – – – – – **Large-scale Pre-training** — GeoPT

Neural v.s. Numerical Simulators





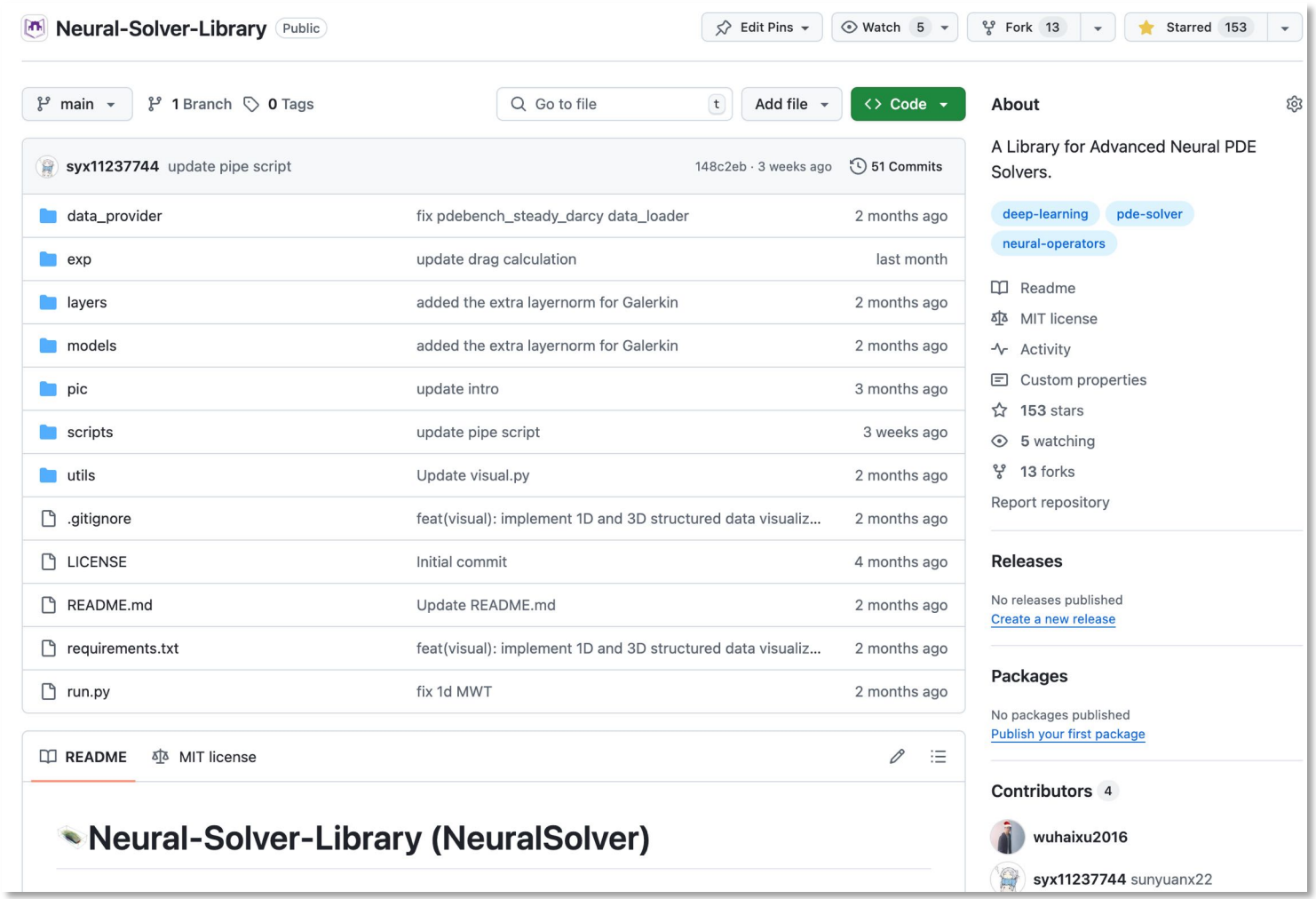
Where we are

GeoPT is **NOT** a physics foundation model.

All the directions in ML community are pursuing the GPT-3 moment.

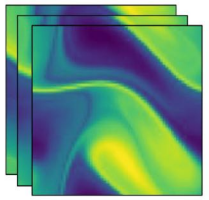
BERT	GPT
• Task-specific fine-tuning	• Zero-/few-shot prompting • Next-token prediction
GeoPT	(?)

Neural-Solver-Library

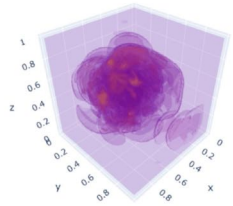


Code Link: *https://github.com/thuml/Neural-Solver-Library*

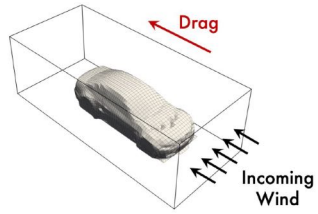
- ✓ 17 different PDE solvers
- ✓ 6 standard benchmarks, PDEBench and design tasks



Task 1: Standard



Task 2: PDEBench



Task 3: ShapeNet Car

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

Takeaways

1. Neural simulator surpasses numerical simulators in scalability

Numerical simulator is a fixed program, which cannot be improved by increasing data, while neural simulator offers a possible way to take benefit from scaling up.

2. Geometry scaling is more like an ML system problem.

In the past five years, ML researchers have realized that the “Bitter Lesson”. Compared to building a new architecture, an equivalent implementation is more reliable.

3. GeoPT provides a way to “bake physics prior from data to parameter”.

In simulation, we can create a clear and isolated scenario for data collection, which offers another way to inform the model with underlying physical laws.

Acknowledgement


Mingsheng Long


Jianmin Wang

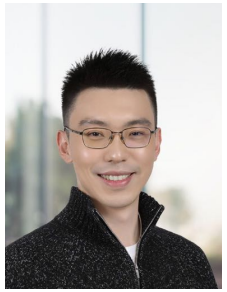

Wojciech Matusik


Kaiming He

NVIDIA

physicsnemo

SIEMENS


Minghao Guo


Hang Zhou


Yuezhou Ma


Huakun Luo


Haonan ShangGuan


Yuanxu Sun


Huikun Weng

From the Transolver Family to GeoPT:

A Scaling Path for Neural Simulators

Haixu Wu

MIT CSAIL

May 12, 2026