A Roadmap to Practical Neural PDE Solvers

Mingsheng Long

School of Software, Tsinghua University

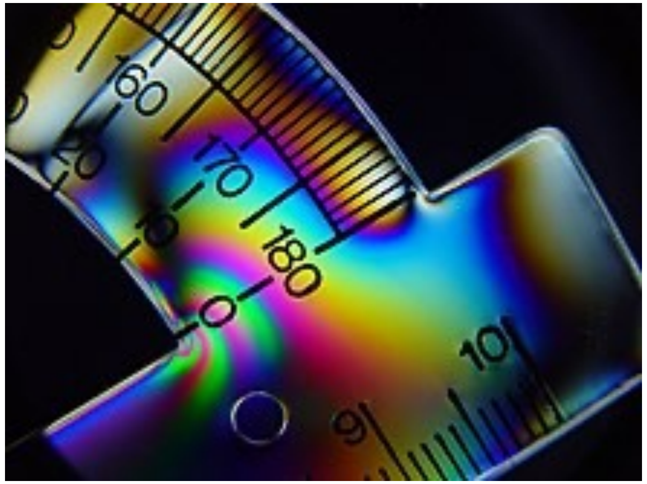
July 2024

清華大學
Tsinghua University

Real-world phenomena

Turbulence

Atmospheric circulation



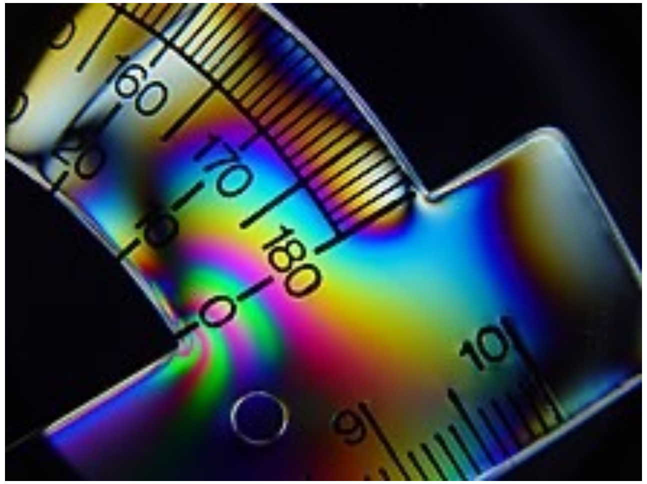
Stress

How to understand the world?

Real-world phenomena

Turbulence

Atmospheric circulation



Stress

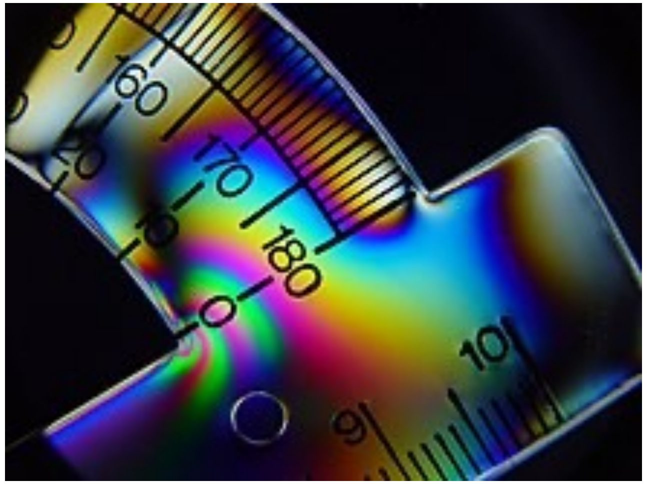
How to understand the world?

Images? Videos?

Real-world phenomena

Turbulence

Atmospheric circulation



Stress

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

Partial Differential Equations (PDEs)

- Fluid physics:

Navier-Stokes Equation for fluid dynamics

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{U}) = 0$$

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

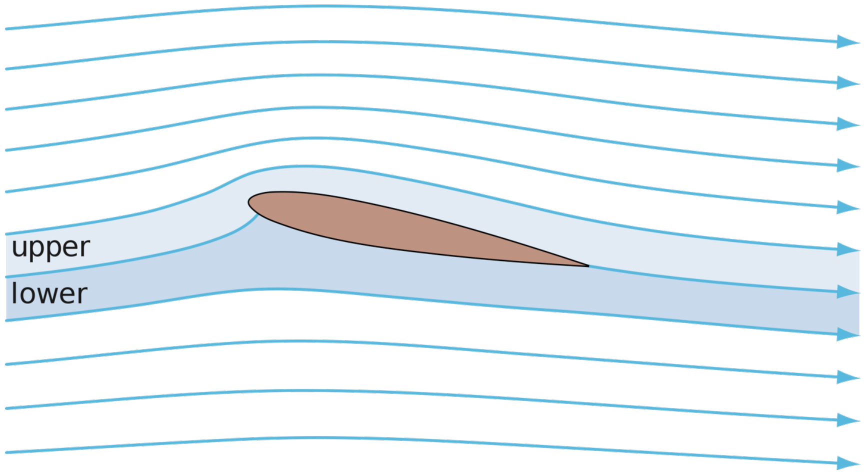
$$\frac{\partial (e + \frac{1}{2} \boldsymbol{U}^2)}{\partial t} + \boldsymbol{U} \cdot \nabla (e + \frac{1}{2} \boldsymbol{U}^2) = \boldsymbol{f} \cdot \boldsymbol{U} + \frac{1}{\rho} \nabla \cdot (\boldsymbol{U} \cdot \boldsymbol{T}_{ij} \boldsymbol{e}_i \boldsymbol{e}_j) + \frac{\lambda}{\rho} \Delta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

- Solid physics:

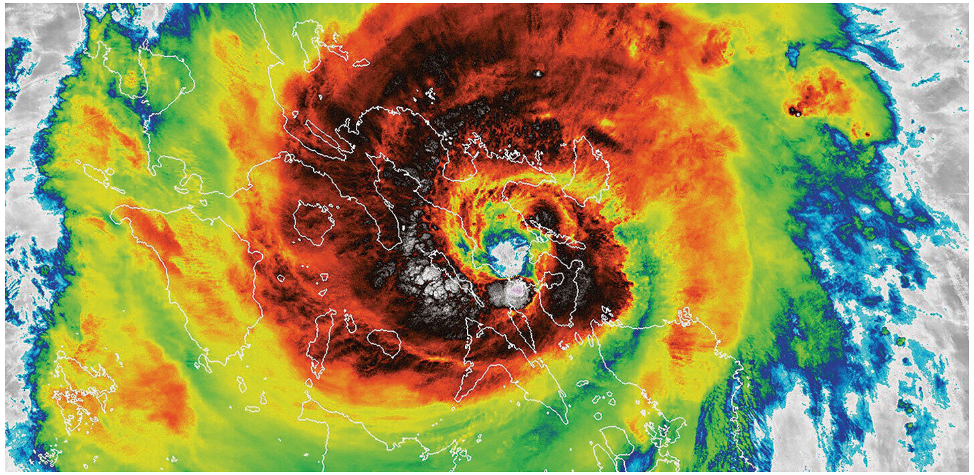
$$\rho^s \frac{\partial^2 \boldsymbol{u}}{\partial t^2} + \nabla \cdot \boldsymbol{\sigma} = 0$$

Inner stress of solid materials

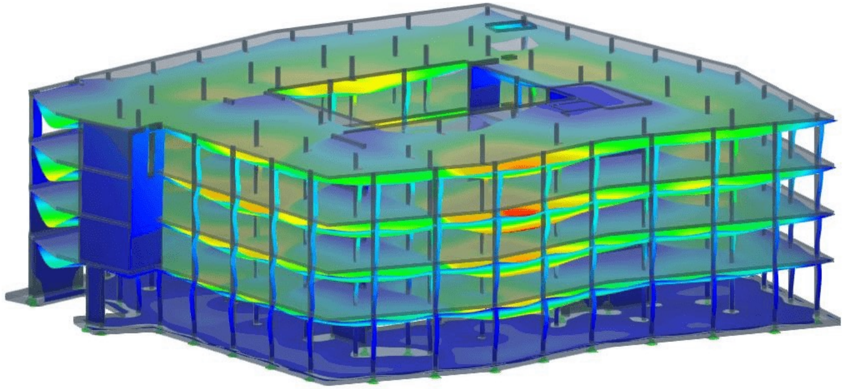
Wide Applications



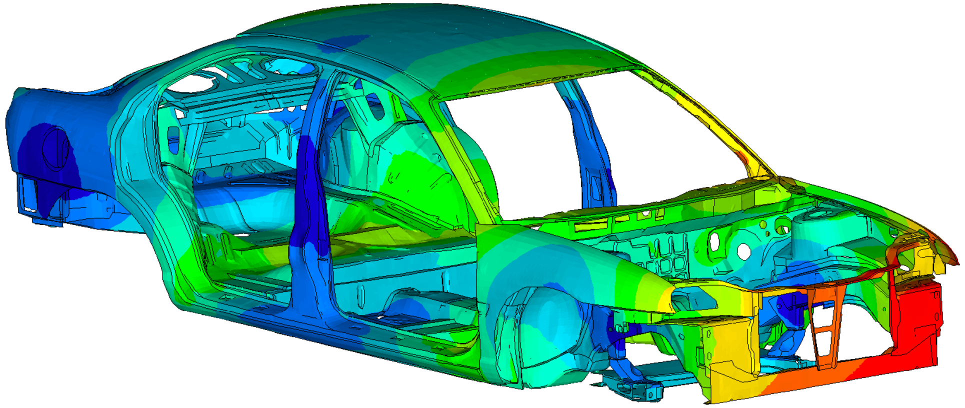
Airfoil design



Weather forecasting



Civil engineering



Vehicle manufacturing

PDE Solvers

Classic Numerical Methods

New Task → FEM, Spectral, etc → Results

- Recalculation for every new sample
- Each round will take hours or even days

Stable but Slow

PDE Solvers

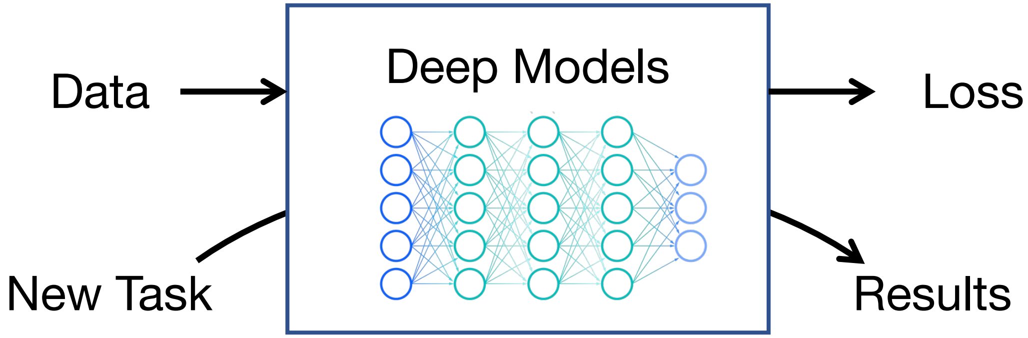
Classic Numerical Methods

New Task → FEM, Spectral, etc → Results

- ➢ Recalculation for every new sample
- ➢ Each round will take hours or even days

Stable but Slow

Neural PDE Solver



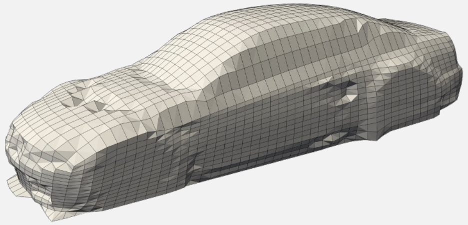
- ➢ Training once, inference a lot
- ➢ Each round needs several seconds

An efficient surrogate tool

(In expectation)

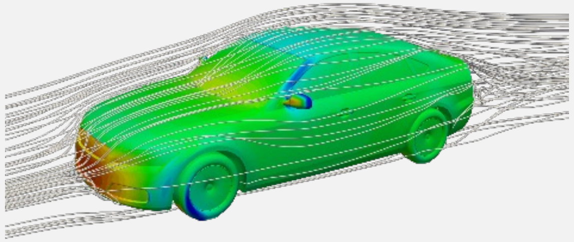
A Roadmap to Practical Neural PDE Solvers

Industrial simulation with CAE



Varied Geometries

Ansys DASSAULT SYSTEMES SIEMENS HEXAGON



Physical Simulation

Neural PDE Solver (Our work)

Q1: High-dimensional Mapping Approximation

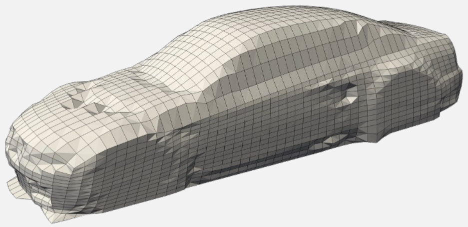
Q2: Large-scale Irregular Meshes

Q3: Generalization among varied PDEs

Deep Models → Latent Spectral Models → Transolver → Unisolver → ...

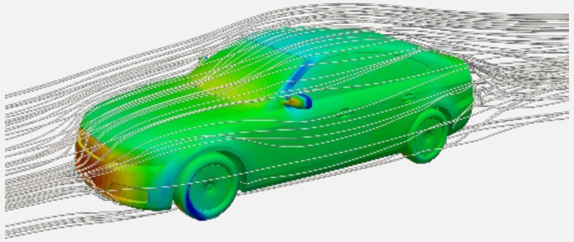
A Roadmap to Practical Neural PDE Solvers

Industrial simulation with CAE



Varied Geometries

Ansys DASSAULT SYSTEMES SIEMENS HEXAGON



Physical Simulation

Neural PDE Solver (Our work)

Q1: High-dimensional Mapping Approximation

Q2: Large-scale Irregular Meshes

Q3: Generalization among varied PDEs

Deep Models — **Latent Spectral Models** — Transolver — Unisolver …

Solving High-Dimensional PDEs with Latent Spectral Models

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

Haixu Wu

Tengge Hu

Huakun Luo

Jianmin Wang

Mingsheng Long

Solving PDEs: Discretization

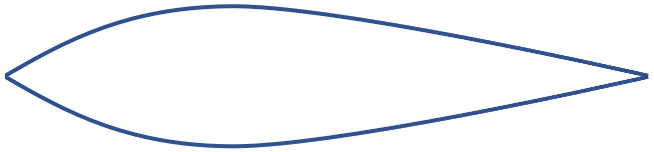
Infinite–dimensional PDE solutions

Discretization

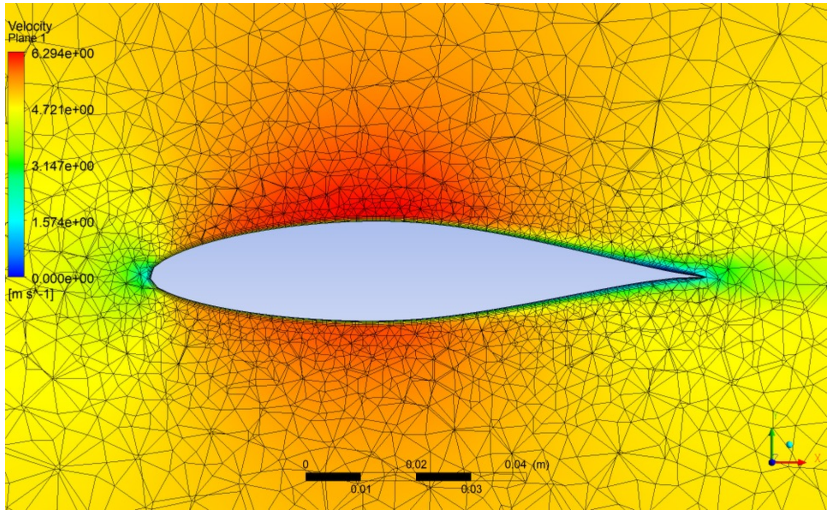
High-dimensional coordinate spaces

$\boldsymbol{x}(\boldsymbol{s}), \boldsymbol{s} \in \mathcal{D}$

$\mathcal{D}$ is the mesh point set



Spatial continuous



Solving PDEs: Discretization

Infinite–dimensional PDE solutions

$\boldsymbol{x}(\boldsymbol{s}), \boldsymbol{s} \in \mathcal{D}$

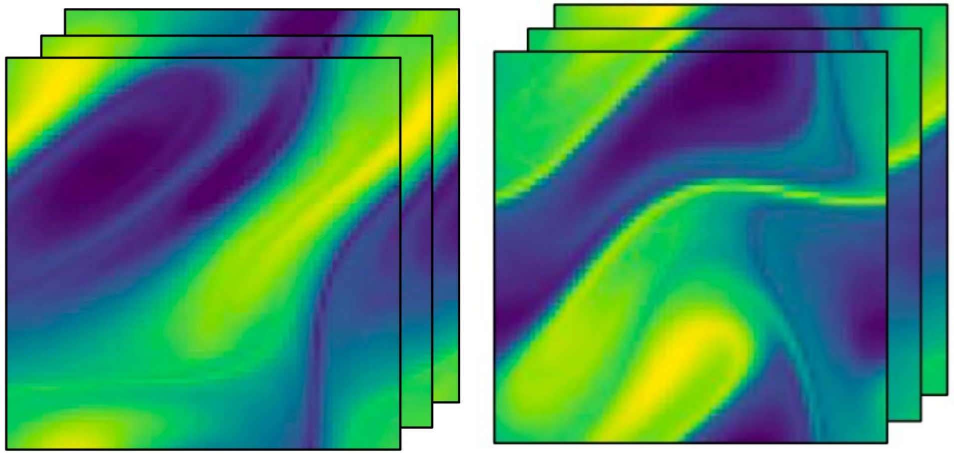
Discretization

High-dimensional coordinate spaces

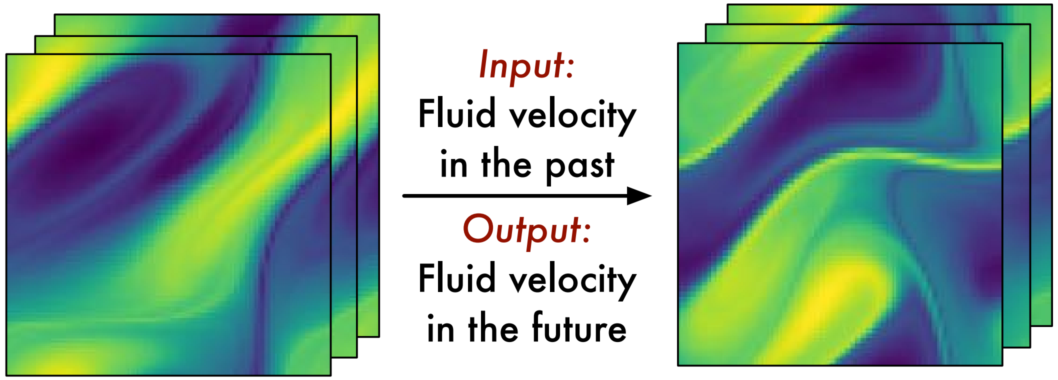
$\mathcal{D}$ is the grid point set

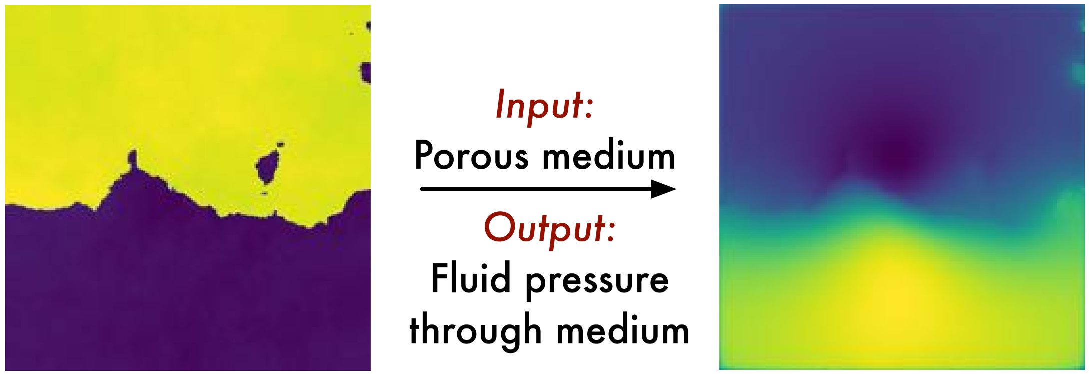
Spatiotemporal Continuous

Navier-Stokes Equation



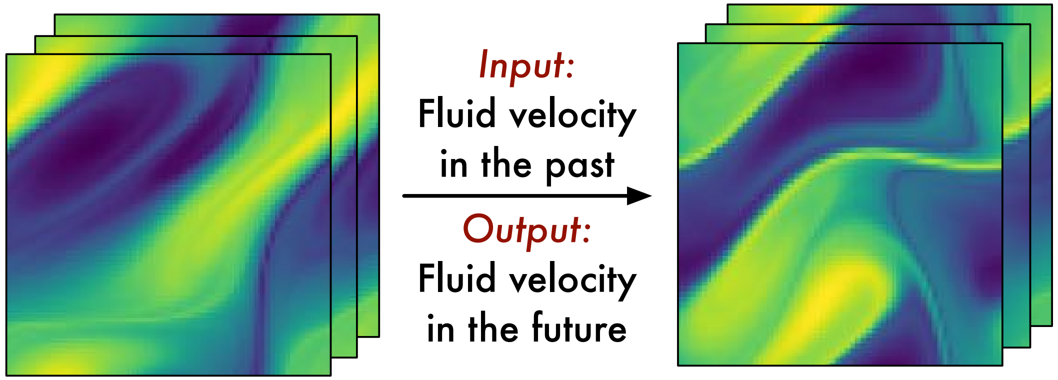
Challenges in Solving High-dimensional PDEs

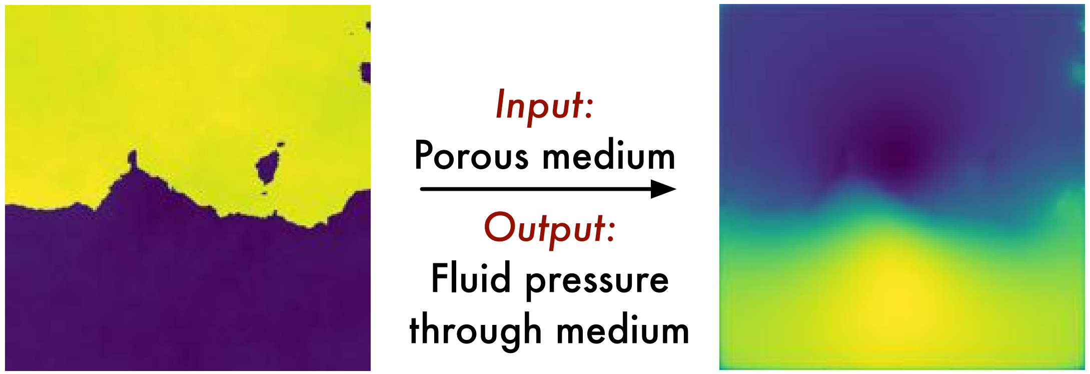




- Curse of dimensionality → Huge computation cost
- Intricate interactions among physical variates of coupled equations → Complex mappings

Challenges in Solving High-dimensional PDEs

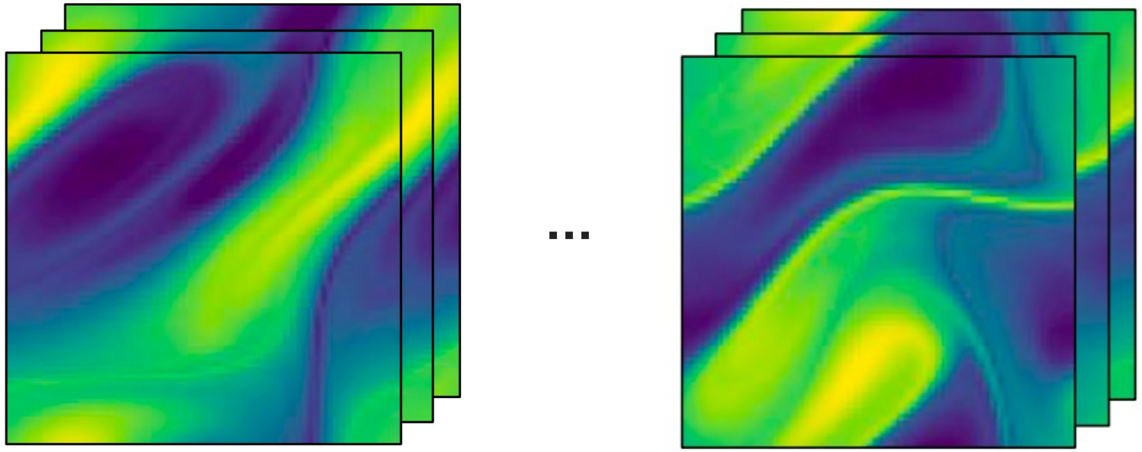




- ➢ Curse of dimensionality → Huge computation cost
- ➢ Intricate interactions among physical variates of coupled equations → Complex mappings

How to efficiently and precisely approximate complex mappings between high-dimensional input-output pairs?

Motivation

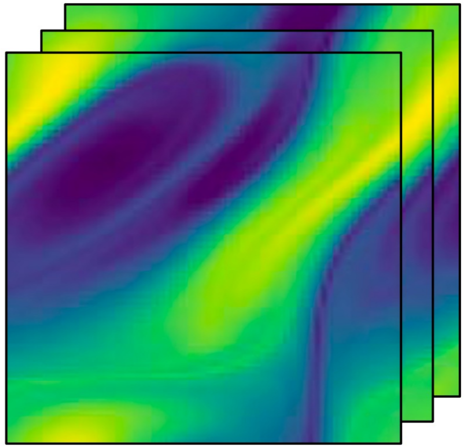


Multitudinous Data

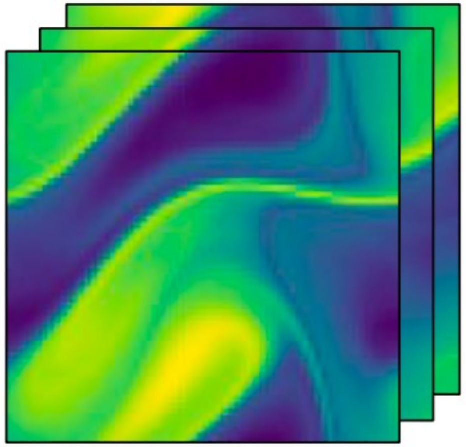
Following the same PDE constraint

Manifold Hypothesis: *Real-world high-dimensional data lie on low-dimensional manifolds embedded within the high-dimensional space.*

Motivation



...



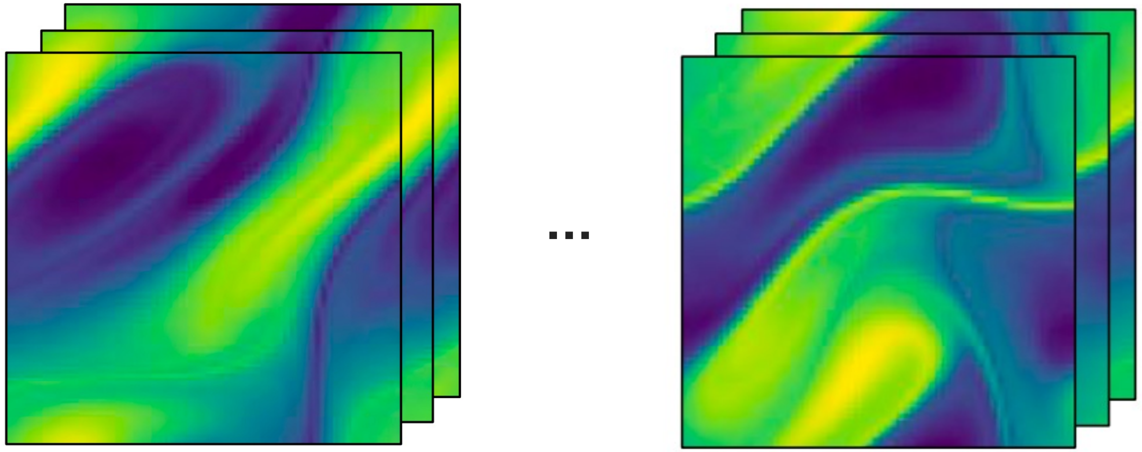
Multitudinous Data

Following the same PDE constraint

Manifold Hypothesis: *Real-world high-dimensional data lie on low-dimensional manifolds embedded within the high-dimensional space.*

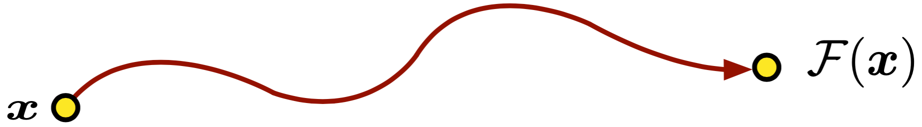
1. High-dimensional data can be projected to a more compact latent space

Motivation



How to approximate

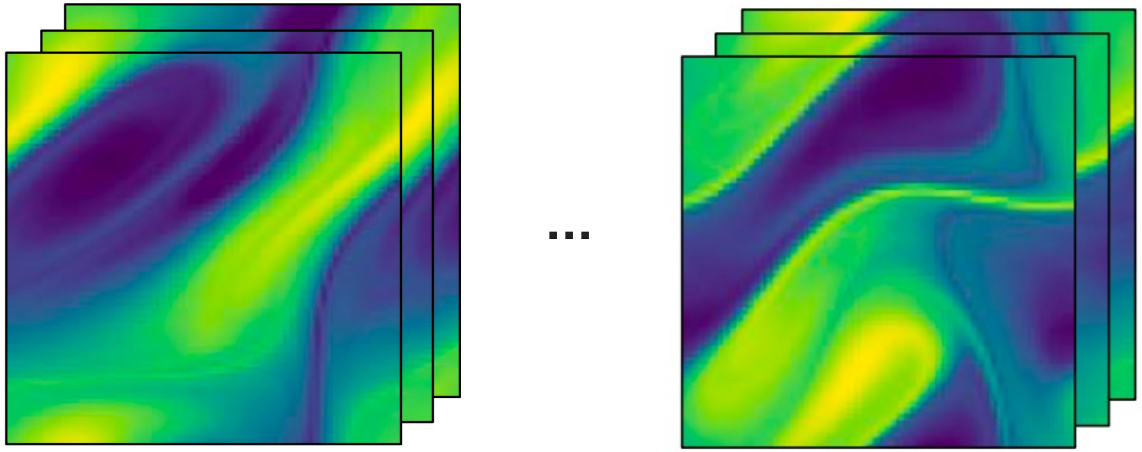
complex input-output mappings?



Previous Methods: Directly approximating with a single deep model

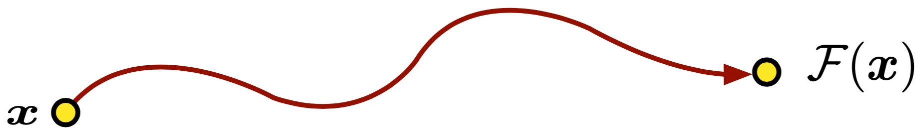
Suffer from optimization problem and limited performance

Motivation



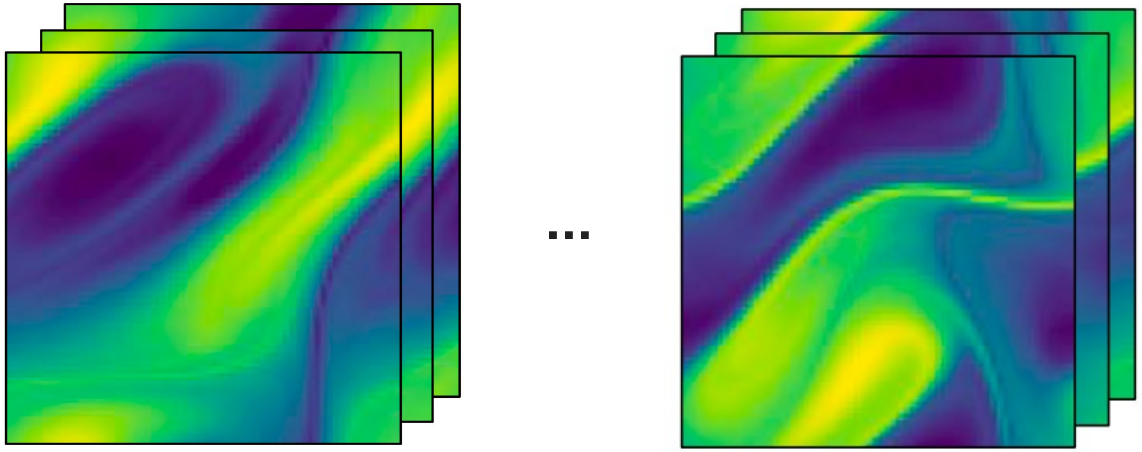
How to approximate

complex input-output mappings?



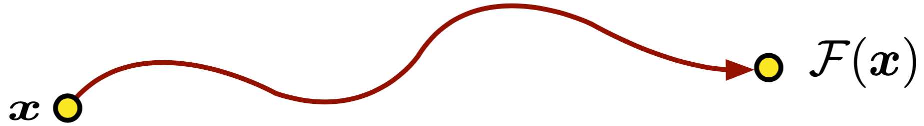
Spectral Methods: *approximate solution f of a certain PDE as a finite sum of N orthogonal basis functions $\{f_1, f_2, \cdots, f_N\}$, that is: $f \approx f^N = \sum_{i=1}^{N} w_i f_i$.*

Motivation



How to approximate

complex input-output mappings?



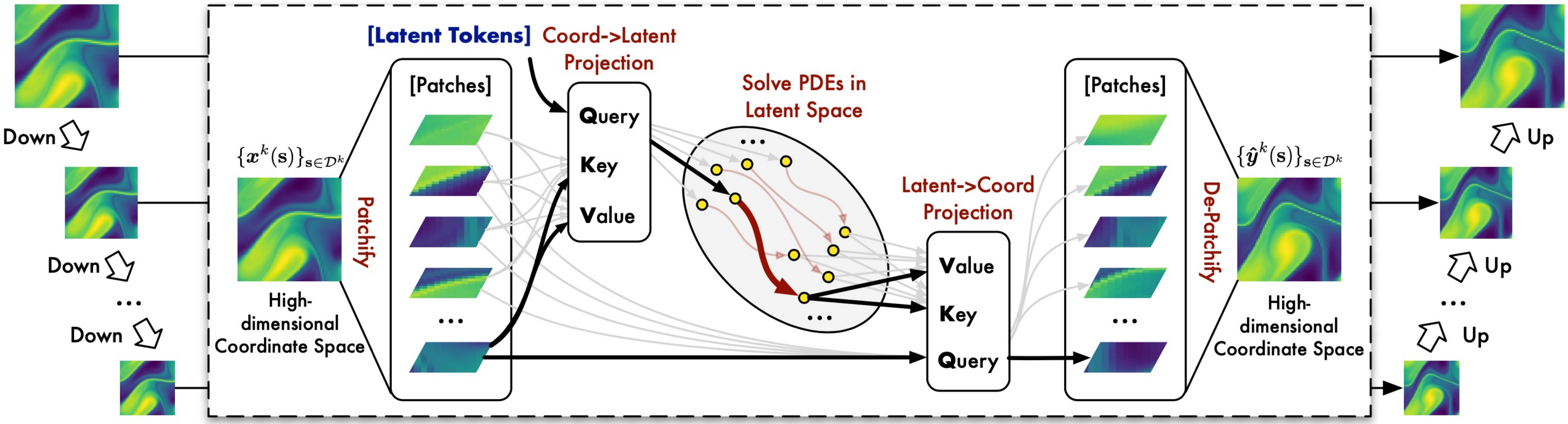
Spectral Methods: *approximate solution f of a certain PDE as a finite sum of N orthogonal basis functions* $\{f_1, f_2, \cdots, f_N\}$, *that is:* $f \approx f^N = \sum_{i=1}^{N} w_i f_i$.

2. Learning multiple basis operators for approximation

Latent Spectral Models (LSM)

	Previous Methods	LSM (ours)
Solving Process	Solving in the **coordinate space** • Huge computation cost • Making input-output mappings extremely complex	Solving in the **latent space** • Efficient computation • Highlight the inherent physics properties
Mapping approximation	Directly learning **a single operator** • Fail in approximating complex mappings • Lack of theoretical guarantee	Learning **multiple basis operators** • Nice approximating and convergence properties under theoretical guarantee

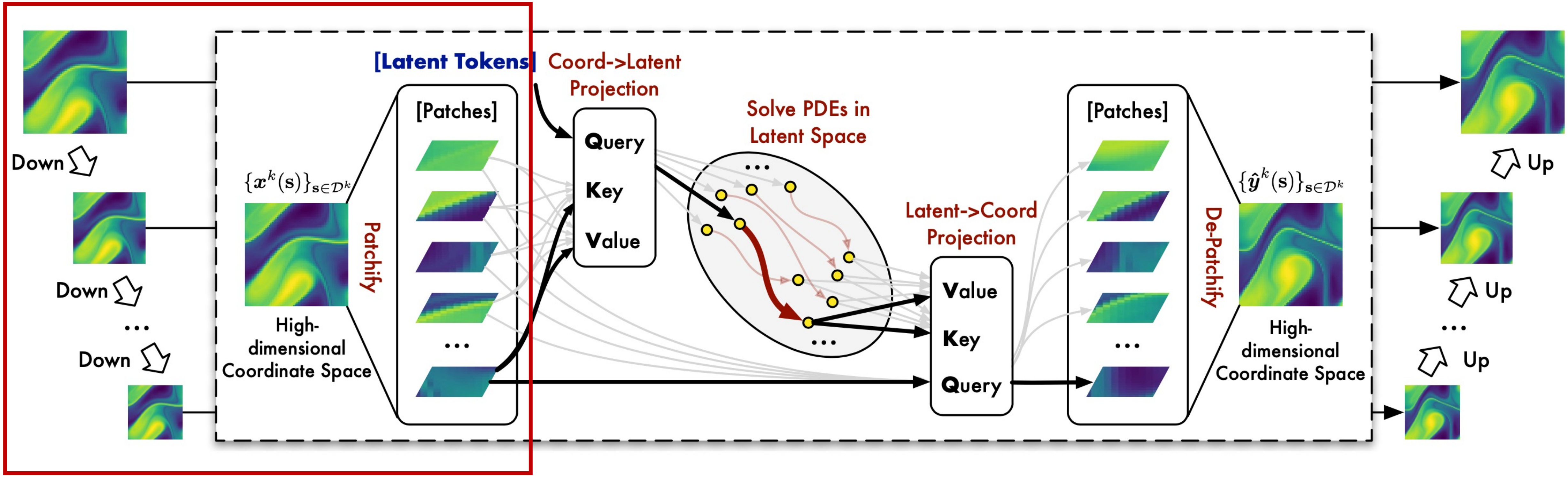
Overall design of LSM



LSM with *Hierarchical Projection Network* and *Neural Spectral Block*

① Coor → Latent ② Solving in the Latent Space ③ Latent → Coor

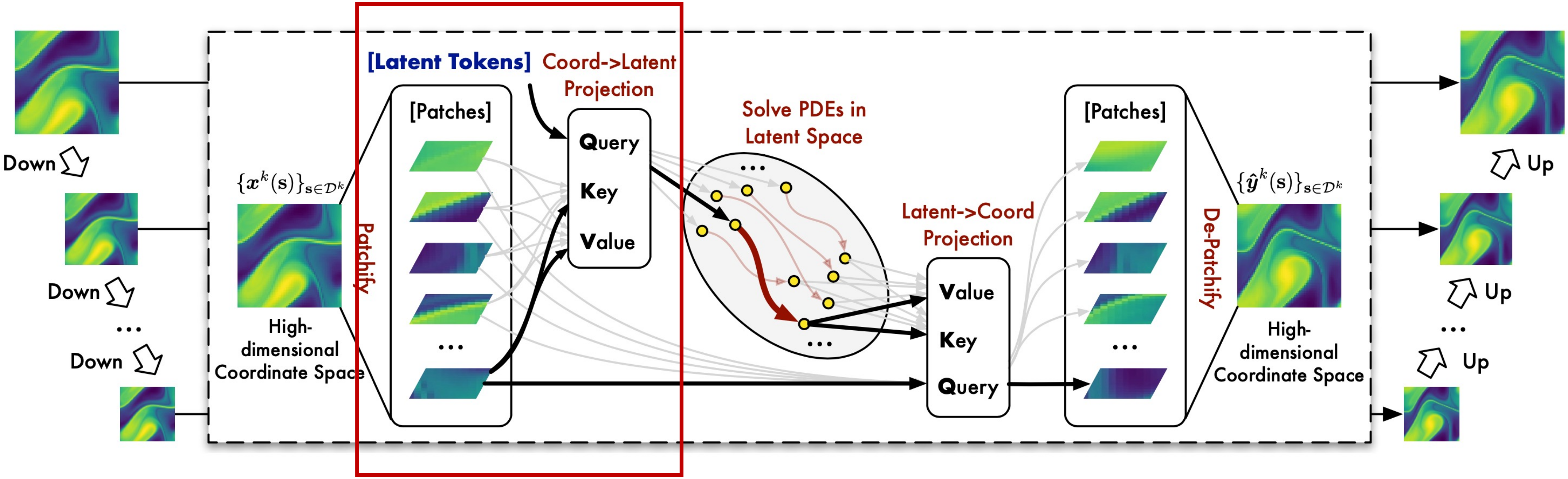
Hierarchical Projection Network



① ***Multiscale patchified architecture*** *→ Solve PDEs in different regions and scales*

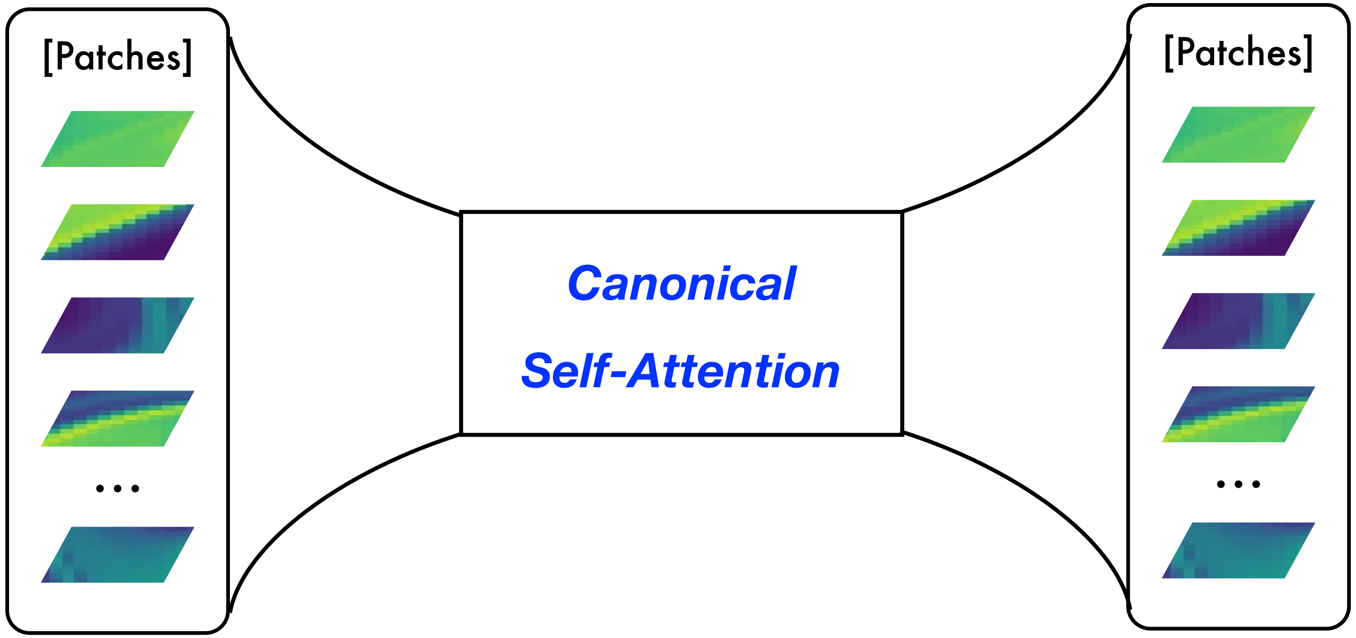
PDEs always present different physical states according to the observed scales and regions.

Hierarchical Projection Network



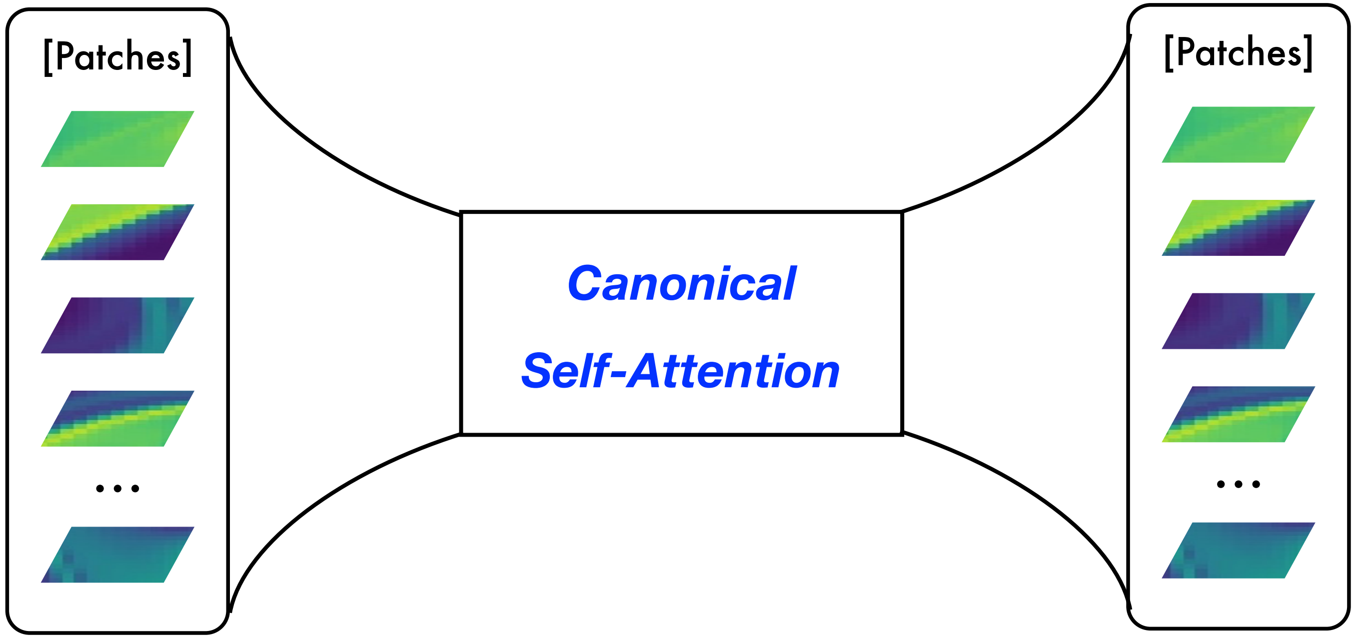
② ***Attention-based projector*** *→ Remove unwieldy coordinate information*

Hierarchical Projection Network



② ***Attention-based projector*** → *Remove unwieldy coordinate information*

Hierarchical Projection Network

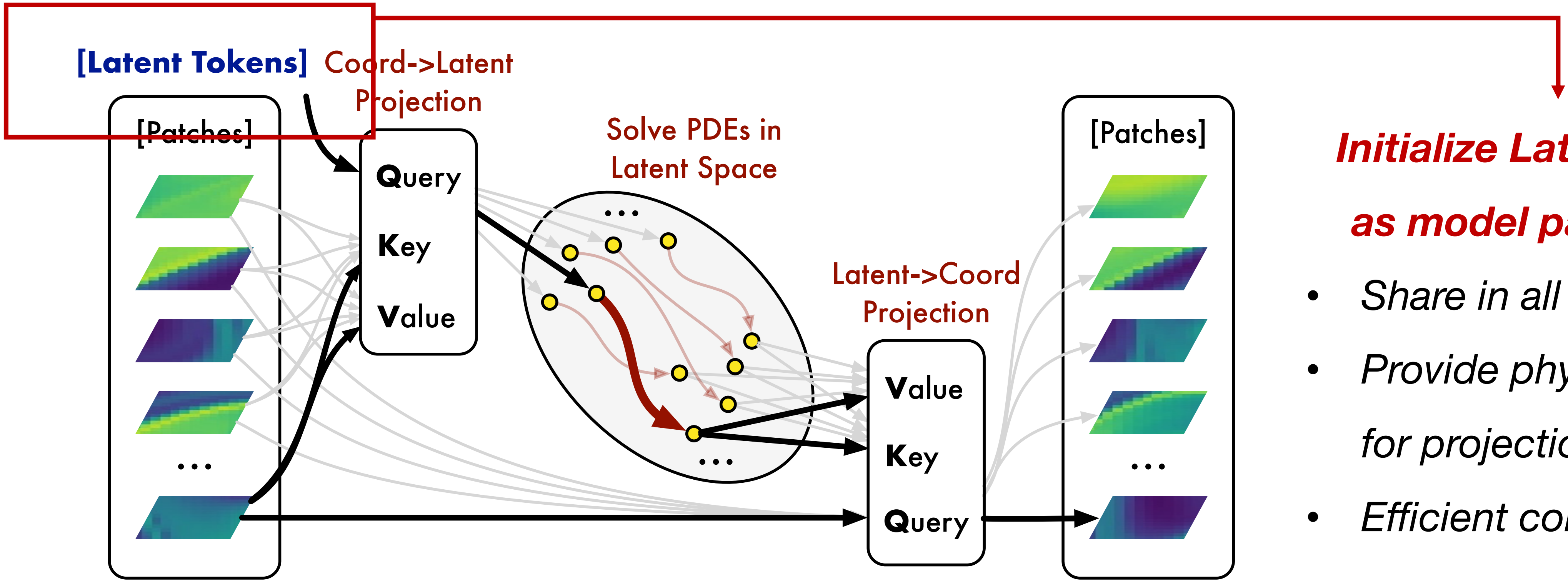


Still in the coordinate space

Undergoing the problems from high-dimensional PDEs

② ***Attention-based projector*** → *Remove unwieldy coordinate information*

Hierarchical Projection Network

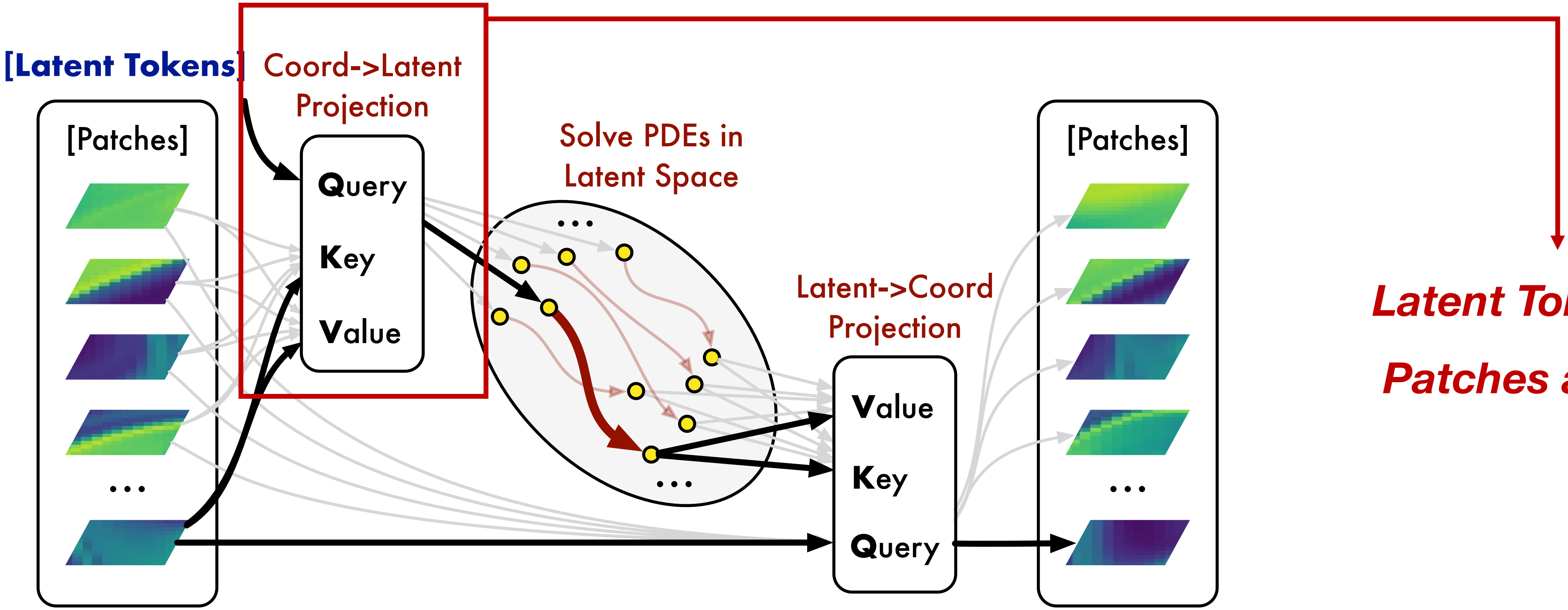


Initialize Latent tokens as model parameters

- *Share in all samples*
- *Provide physical prompt for projection*
- *Efficient computation*

② ***Attention-based projector*** → *Remove unwieldy coordinate information*

Hierarchical Projection Network

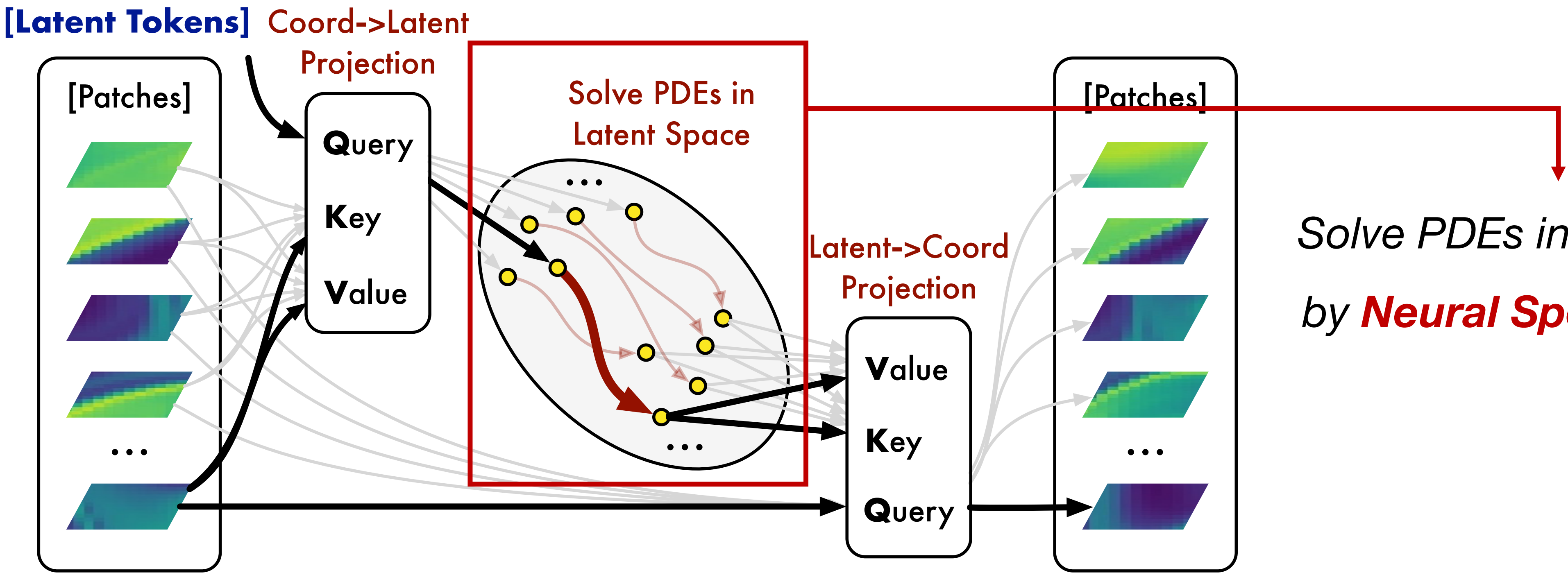


Latent Token as Q

Patches as K & V

$$\mathbf{T}_{\boldsymbol{x},i} = \mathbf{T}_i + \sum_{\mathbf{s}\in\mathcal{D}} \frac{\mathrm{Sim}\left(\mathbf{T}_i, \boldsymbol{x}(\mathbf{s})\mathbf{W}_{\mathrm{K}}\right)}{\sum_{\mathbf{s}'\in\mathcal{D}} \mathrm{Sim}\left(\mathbf{T}_i, \boldsymbol{x}(\mathbf{s}')\mathbf{W}_{\mathrm{K}}\right)} \left(\boldsymbol{x}(\mathbf{s})\mathbf{W}_{\mathrm{V}}\right)$$

Hierarchical Projection Network

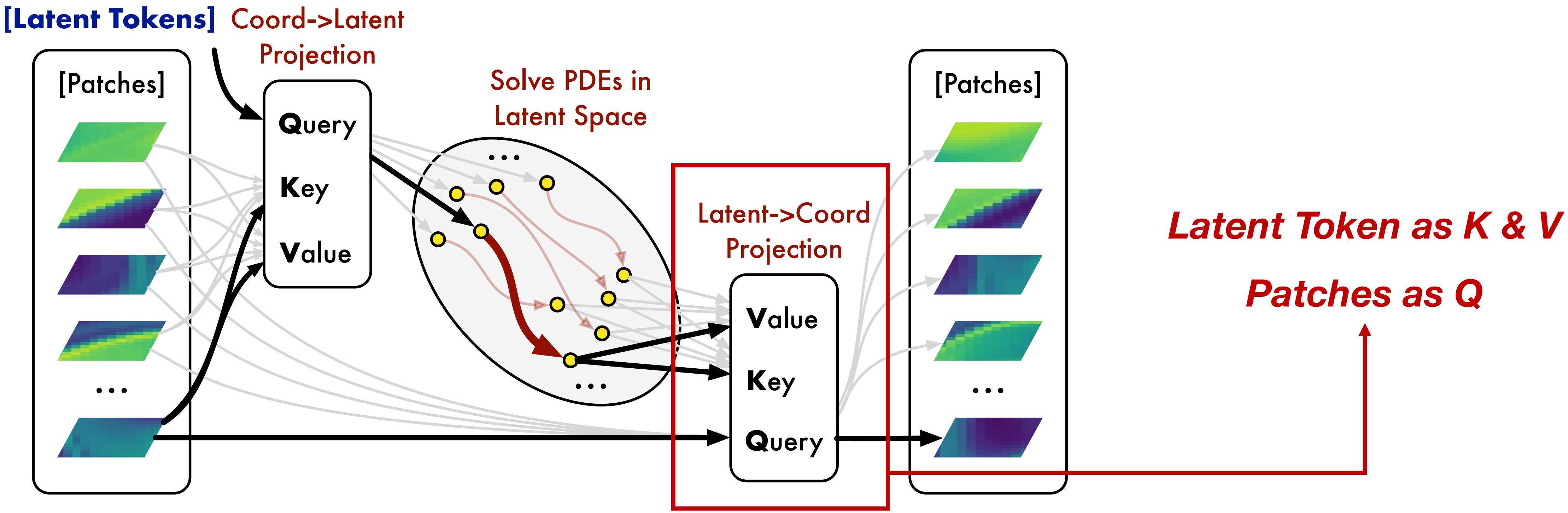


Solve PDEs in latent space

by ***Neural Spectral Block***

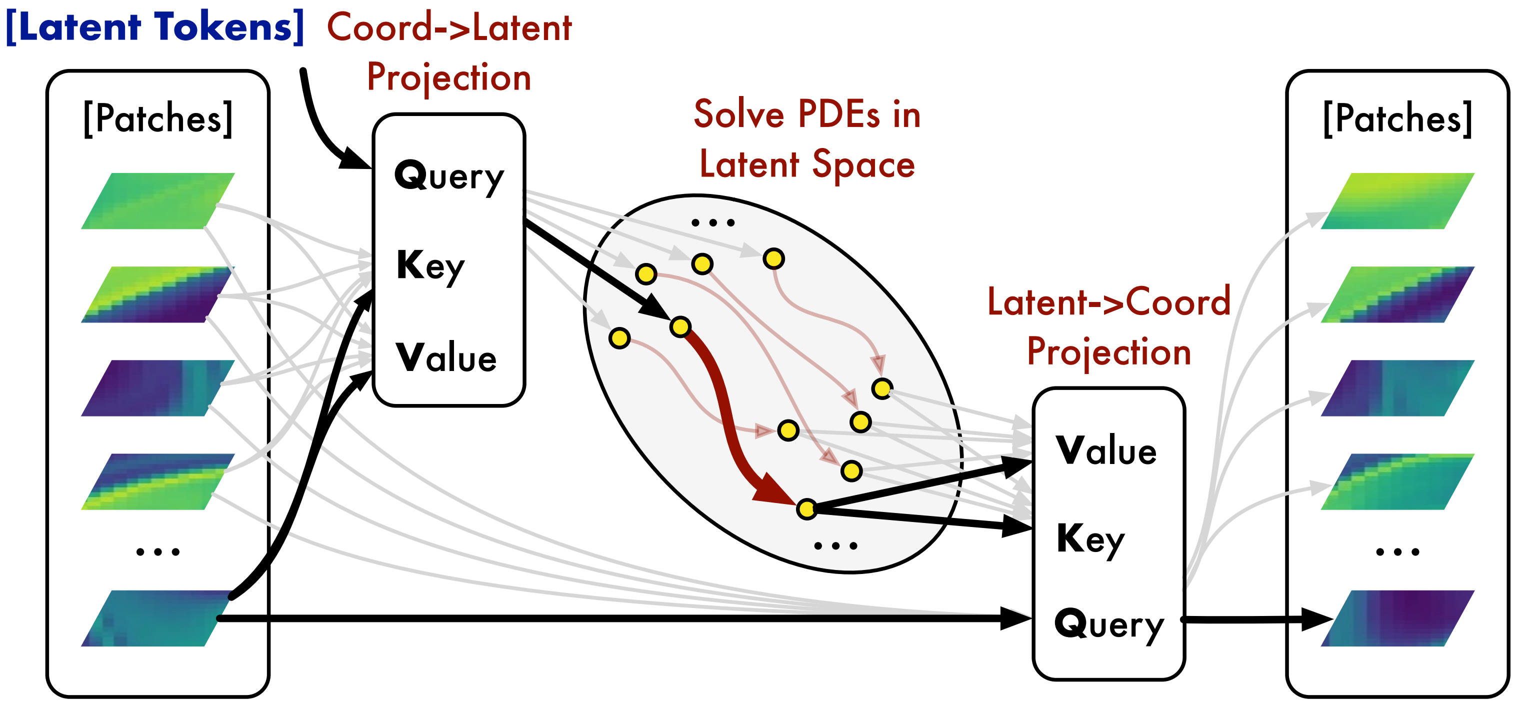
$$\{\mathbf{T}^k_{\boldsymbol{y},i,j}\}_{i=1}^{C} = \mathrm{Solve}\left(\{\mathbf{T}^k_{\boldsymbol{x},i,j}\}_{i=1}^{C}\right)$$

Hierarchical Projection Network



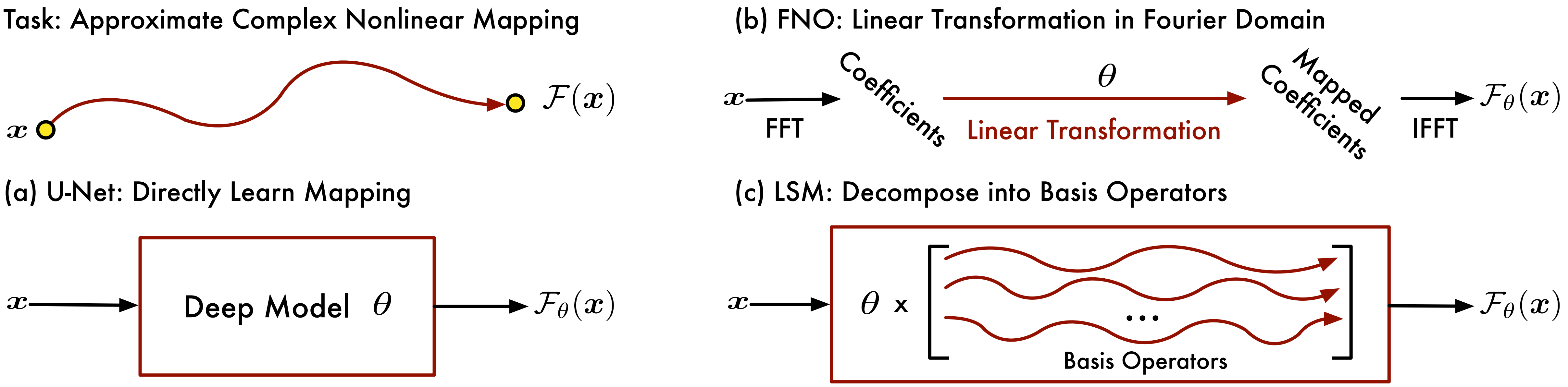
$$\widehat{\boldsymbol{y}}(\mathbf{s}) = \boldsymbol{x}(\mathbf{s}) + \sum_{i=1}^{C} \frac{\mathrm{Sim}\left(\boldsymbol{x}(\mathbf{s}), \mathbf{T}_{\boldsymbol{y},i}\mathbf{W}'_{\mathrm{K}}\right)}{\sum_{i'=1}^{C} \mathrm{Sim}\left(\boldsymbol{x}(\mathbf{s}), \mathbf{T}_{\boldsymbol{y},i'}\mathbf{W}'_{\mathrm{K}}\right)} (\mathbf{T}_{\boldsymbol{y},i}\mathbf{W}'_{\mathrm{V}})$$

Hierarchical Projection Network



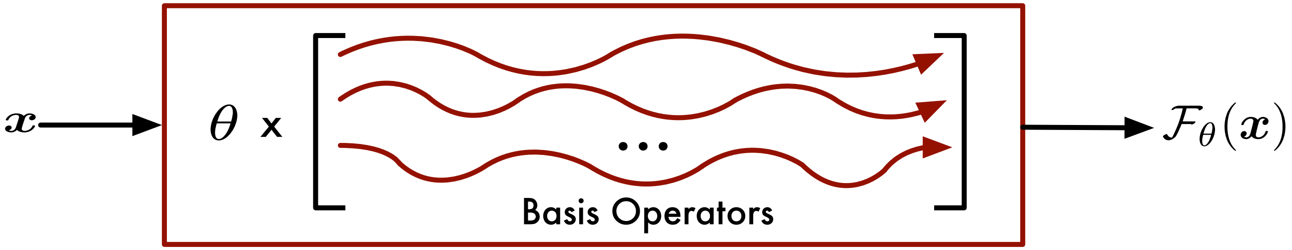
1. ***Linear complexity projection, more efficient computation***
2. ***Highlight the inherent properties of high-dimensional data***
3. ***Benefit the model convergence properties***

Neural Spectral Block



LSM approximates complex mappings by **learning multiple basis operators**

Neural Spectral Block

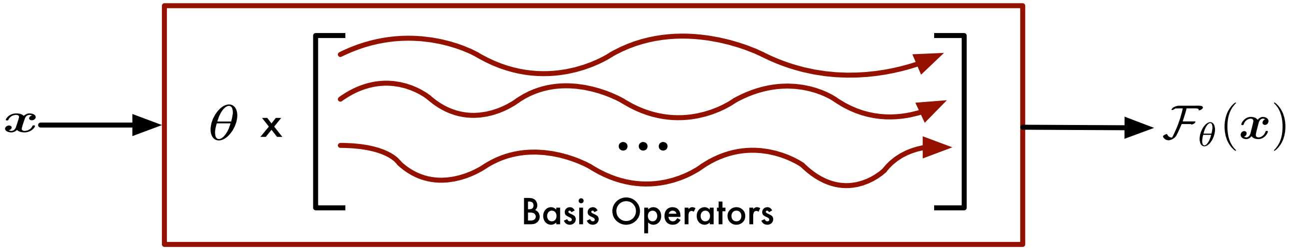


$$\mathcal{F}_{\theta_{\text{Solve}}} = \sum_{i=1}^{N} w_i \mathcal{F}_{\theta_{\text{Solve},i}}$$

We select trigonometric basis operators

$$\mathcal{F}_{\theta_{\text{Solve},(2k-1)}}\left(\boldsymbol{t}_{\boldsymbol{x}}(\mathbf{s})\right) = \sin\left(k\boldsymbol{t}_{\boldsymbol{x}}(\mathbf{s})\right)$$
$$\mathcal{F}_{\theta_{\text{Solve},(2k)}}\left(\boldsymbol{t}_{\boldsymbol{x}}(\mathbf{s})\right) = \cos\left(k\boldsymbol{t}_{\boldsymbol{x}}(\mathbf{s})\right)$$

Neural Spectral Block



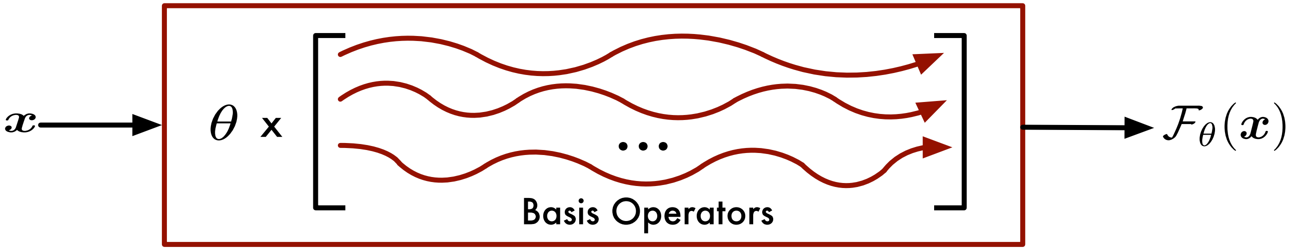
$$\mathcal{F}_{\theta_{\text{Solve}}} = \sum_{i=1}^{N} w_i \mathcal{F}_{\theta_{\text{Solve},i}}$$

$\{\omega_1, \cdots, \omega_N\}$ are model parameters

Will be optimized to satisfy the PDEs better,

during the training process, namely solving PDEs

Neural Spectral Block



$$\mathcal{F}_{\theta_{\text{Solve}}} = \sum_{i=1}^{N} w_i \mathcal{F}_{\theta_{\text{Solve},i}}$$

Neural spectral block is applied to

projected latent tokens of **all the patches in multiple scales**

$$\mathbf{T}_{\boldsymbol{y}} = \mathbf{T}_{\boldsymbol{x}} + \mathbf{w}_0 + \mathbf{w}_{\sin} \begin{bmatrix} \sin(\mathbf{T}_{\boldsymbol{x}}) \\ \vdots \\ \sin(\frac{N}{2}\mathbf{T}_{\boldsymbol{x}}) \end{bmatrix} + \mathbf{w}_{\cos} \begin{bmatrix} \cos(\mathbf{T}_{\boldsymbol{x}}) \\ \vdots \\ \cos(\frac{N}{2}\mathbf{T}_{\boldsymbol{x}}) \end{bmatrix}$$

Theoretical analysis

Convergence of Trigonometric Approximation in High-dimensional Space:

Let $\boldsymbol{f}: \mathbb{R}^M \rightarrow \mathbb{R}^M$ be a 2π-periodic function w.r.t. the variable on each dimension, where $\boldsymbol{f} \in L_p([-\pi, \pi)^M), M \geq 2, 1 \leq p \leq \infty$ and $p \neq 2$. For $\boldsymbol{f}$ defined on the ***M-dimension space****, its trigonometric approximation $\boldsymbol{f}^N$ is defined as:*

$$\boldsymbol{f}^N(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbb{Z}^M, |\mathbf{k}| \leq N} \left(\frac{1}{2\pi} \int_{[-\pi, \pi)^M} \boldsymbol{f}(\mathbf{t}) e^{-i\mathbf{k}\mathbf{t}} \mathrm{d}\mathbf{t} \right) e^{i\mathbf{k}\mathbf{x}},$$

if $\boldsymbol{f}$ satisfies the Lipschitz condition, then there exists a constant K, such that

$$\left|\left|\boldsymbol{f} - \boldsymbol{f}^N\right|\right| \leq K N^{(M-1)\left|\frac{1}{2} - \frac{1}{p}\right| - 1}.$$

Slow Convergence Rate in High-dimensional Space

Theoretical analysis

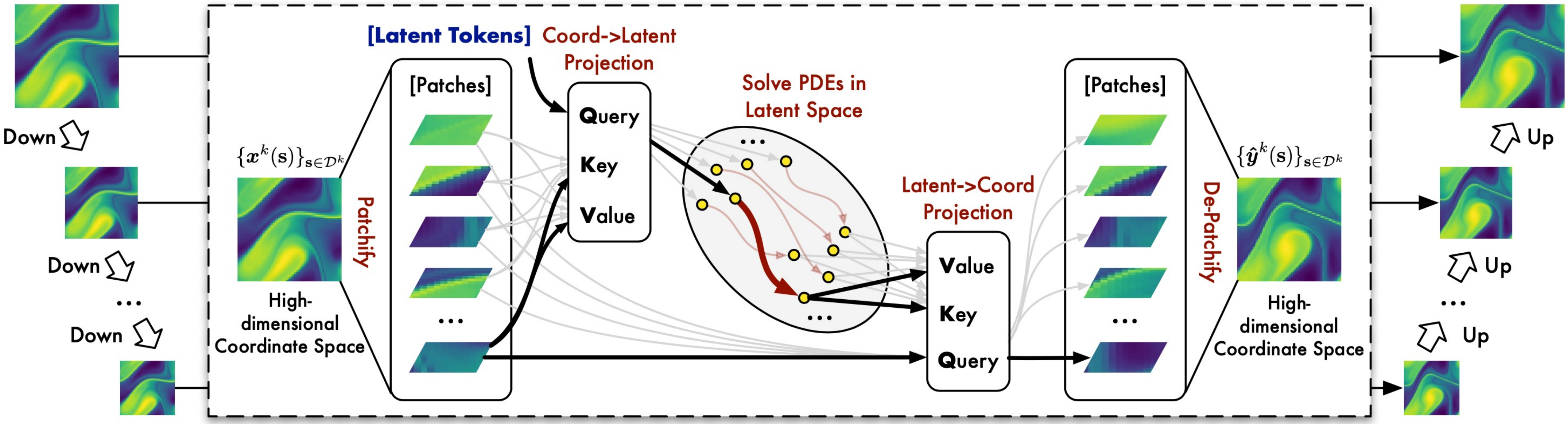
Approximation and Convergence Properties of Neural Spectral Block

(trigonometric approximation with residual): Given $f: [0, \pi] \to \mathbb{R}$, if f satisfies the Lipschitz condition, there is a choice of model parameters such that the approximation f^N defined in neural spectral block can uniformly converge to f with the speed as follows:

$$\left\| f - f^N(\boldsymbol{x}) \right\| \leq K \frac{\ln N}{N}, \forall x \in [0, \pi].$$

Projecting M-dimension data into **independent latent tokens** brings favorable convergence speed of Neural Spectral Block.

Overall design of LSM



LSM with *Hierarchical Projection Network* and *Neural Spectral Block*

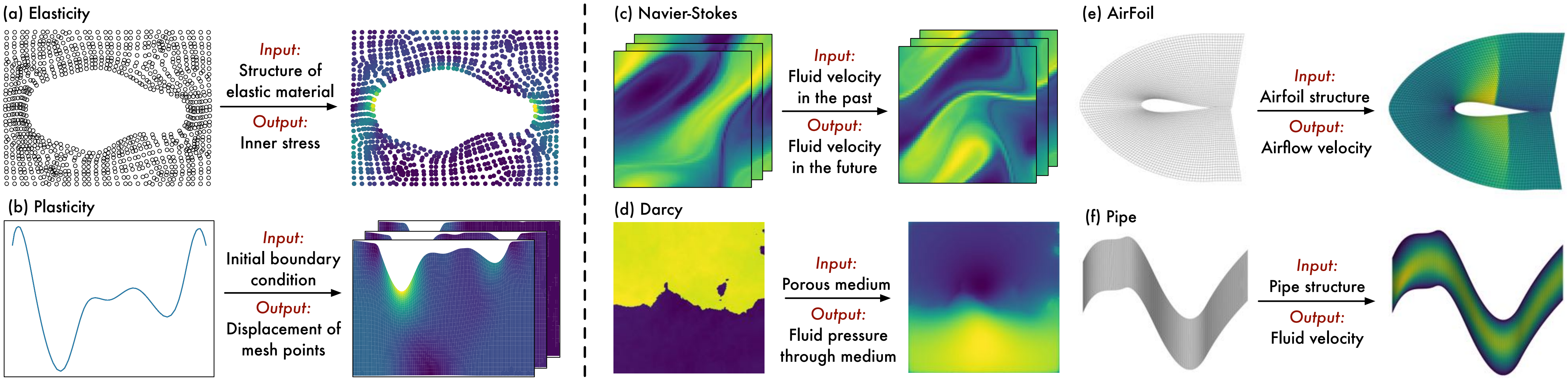
① Coor → Latent ② Solving in the Latent Space ③ Latent → Coor

Experiments

Physics	Benchmarks	Geometry	#Dim
Solid	Elasticity-P	Point Cloud	2D
	Elasticity-G	Regular Grid	2D
	Plasticity	Structured Mesh	3D
Fluid	Navier–Stokes	Regular Grid	3D
	Darcy	Regular Grid	2D
	Airfoil	Structured mesh	2D
	Pipe	Structured mesh	2D

Seven typical PDE solving tasks, covering both fluid and solid physis, various geometrics.

PDE-governed Tasks



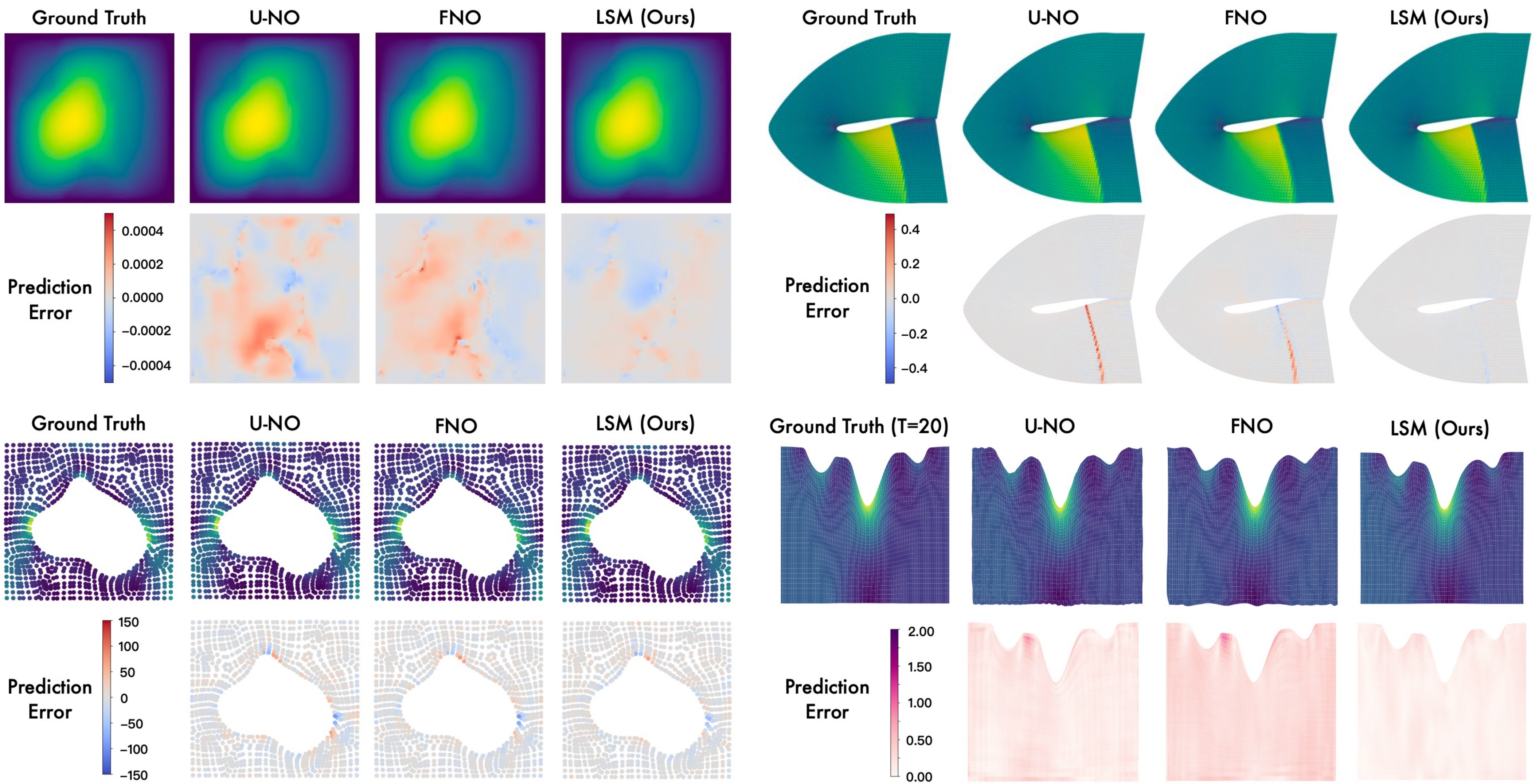
Approximate complex input-output mappings with deep models

Main Results

Model	Solid Physics*			Fluid Physics†			
	Elasticity-P ‡	Elasticity-G	Plasticity	Navier–Stokes	Darcy	Airfoil	Pipe
U-Net (2015)	0.0235	0.0531	0.0051	0.1982	0.0080	0.0079	0.0065
ResNet (2016)	0.0262	0.0843	0.0233	0.2753	0.0587	0.0391	0.0120
TF-Net (2019)	/	/	/	0.1801	/	/	/
Swin (2021)	0.0283	0.0819	0.0170	0.2248	0.0397	0.0270	0.0109
DeepONet (2021)	0.0965	0.0900	0.0135	0.2972	0.0588	0.0385	0.0097
FNO (2021)	<u>0.0229</u>	0.0508	0.0074	0.1556	0.0108	0.0138	0.0067
U-FNO (2021)	0.0239	0.0480	0.0039	0.2231	0.0183	0.0269	<u>0.0056</u>
WMT (2021)	0.0359	0.0520	0.0076	<u>0.1541</u>	0.0082	0.0075	0.0077
Galerkin (2021)	0.0240	0.1681	0.0120	0.2684	0.0170	0.0118	0.0098
SNO (2022)	0.0390	0.0987	0.0070	0.2568	0.0495	0.0893	0.0294
U-NO (2022)	0.0258	<u>0.0469</u>	<u>0.0034</u>	0.1713	0.0113	0.0078	0.0100
HT-Net (2022)	0.0372	0.0472	0.0333	0.1847	0.0079	<u>0.0065</u>	0.0059
F-FNO (2023)	0.0263	0.0475	0.0047	0.2322	<u>0.0077</u>	0.0078	0.0070
KNO (2023a)	/	/	/	0.2023	/	/	/
LSM	**0.0218**	**0.0408**	**0.0025**	**0.1535**	**0.0065**	**0.0059**	**0.0050**
Promotion	4.8%	13.0%	26.5%	0.4%	15.6%	9.2%	10.7%

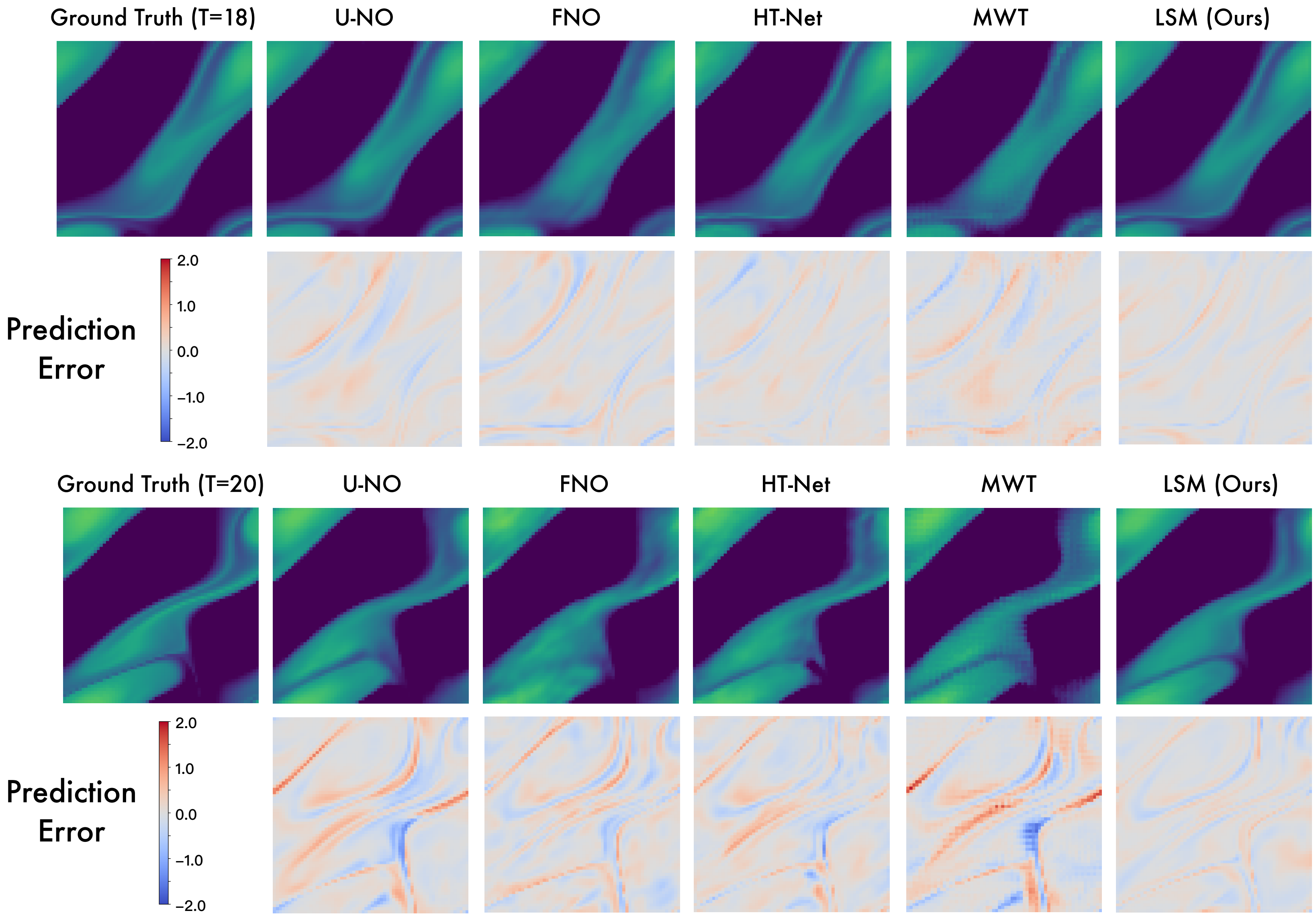
LSM achieves consistent SOTA and surpasses previous 14 baselines with 11.5% error reduction.

Showcases



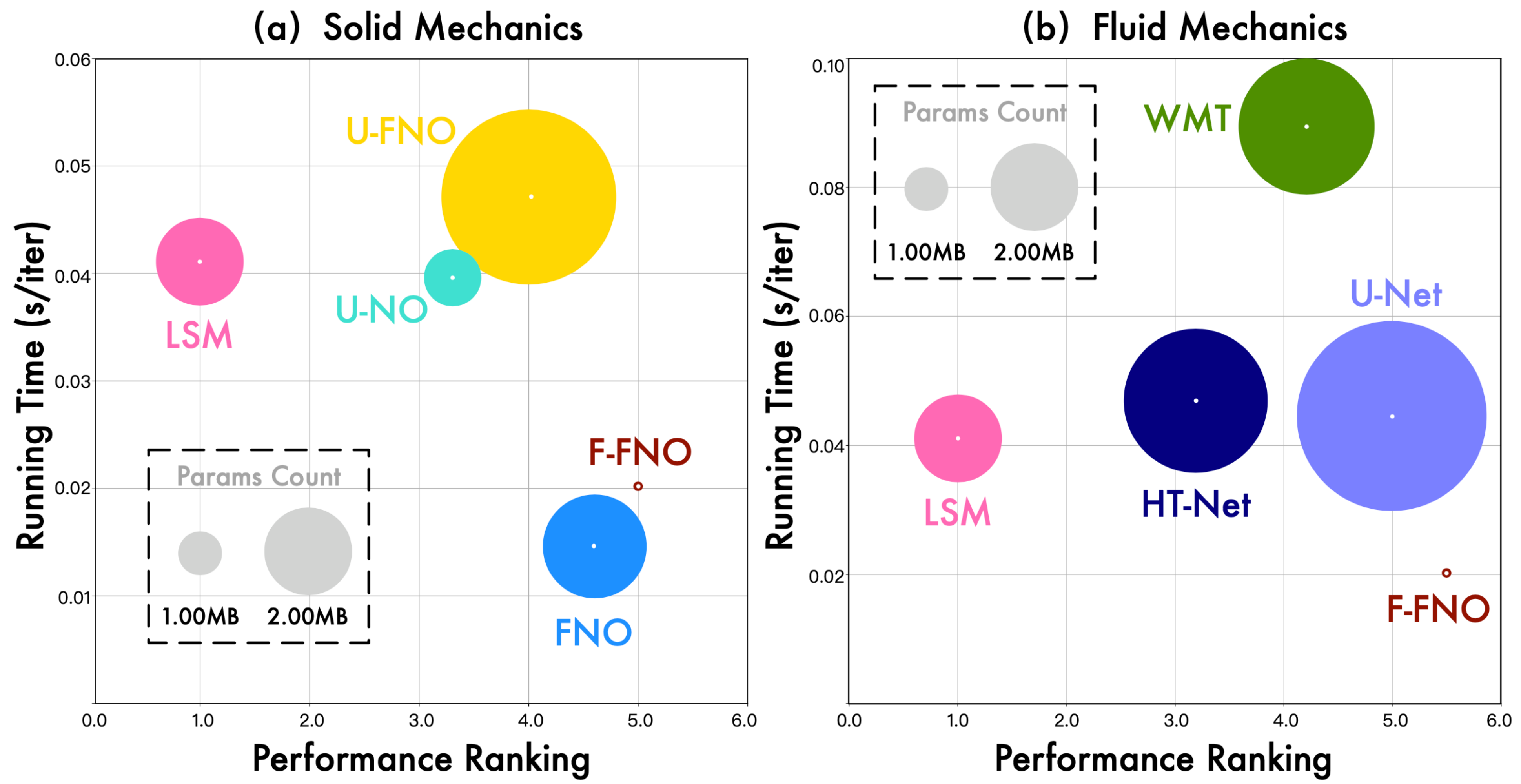
Ground Truth
U-NO
FNO
LSM (Ours)
Prediction Error
0.0004
0.0002
0.0000
−0.0002
−0.0004
Ground Truth
U-NO
FNO
LSM (Ours)
Prediction Error
0.4
0.2
0.0
−0.2
−0.4
Ground Truth
U-NO
FNO
LSM (Ours)
Prediction Error
150
100
50
0
−50
−100
−150
Ground Truth (T=20)
U-NO
FNO
LSM (Ours)
Prediction Error
2.00
1.50
1.00
0.50
0.00

Showcases



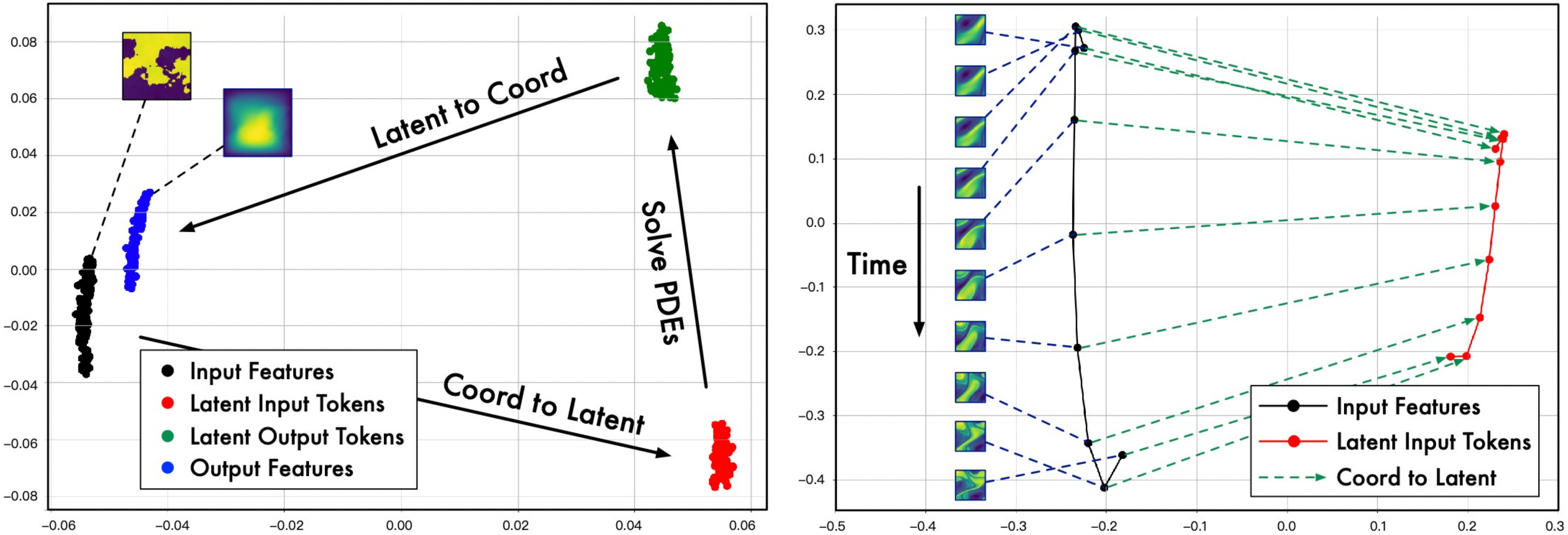
Ground Truth (T=18)
U-NO
FNO
HT-Net
MWT
LSM (Ours)
Prediction Error
2.0
1.0
0.0
−1.0
−2.0
Ground Truth (T=20)
U-NO
FNO
HT-Net
MWT
LSM (Ours)
Prediction Error
2.0
1.0
0.0
−1.0
−2.0

Efficiency



Favorable trade-off between performance and efficiency.

Solving Process Visualization

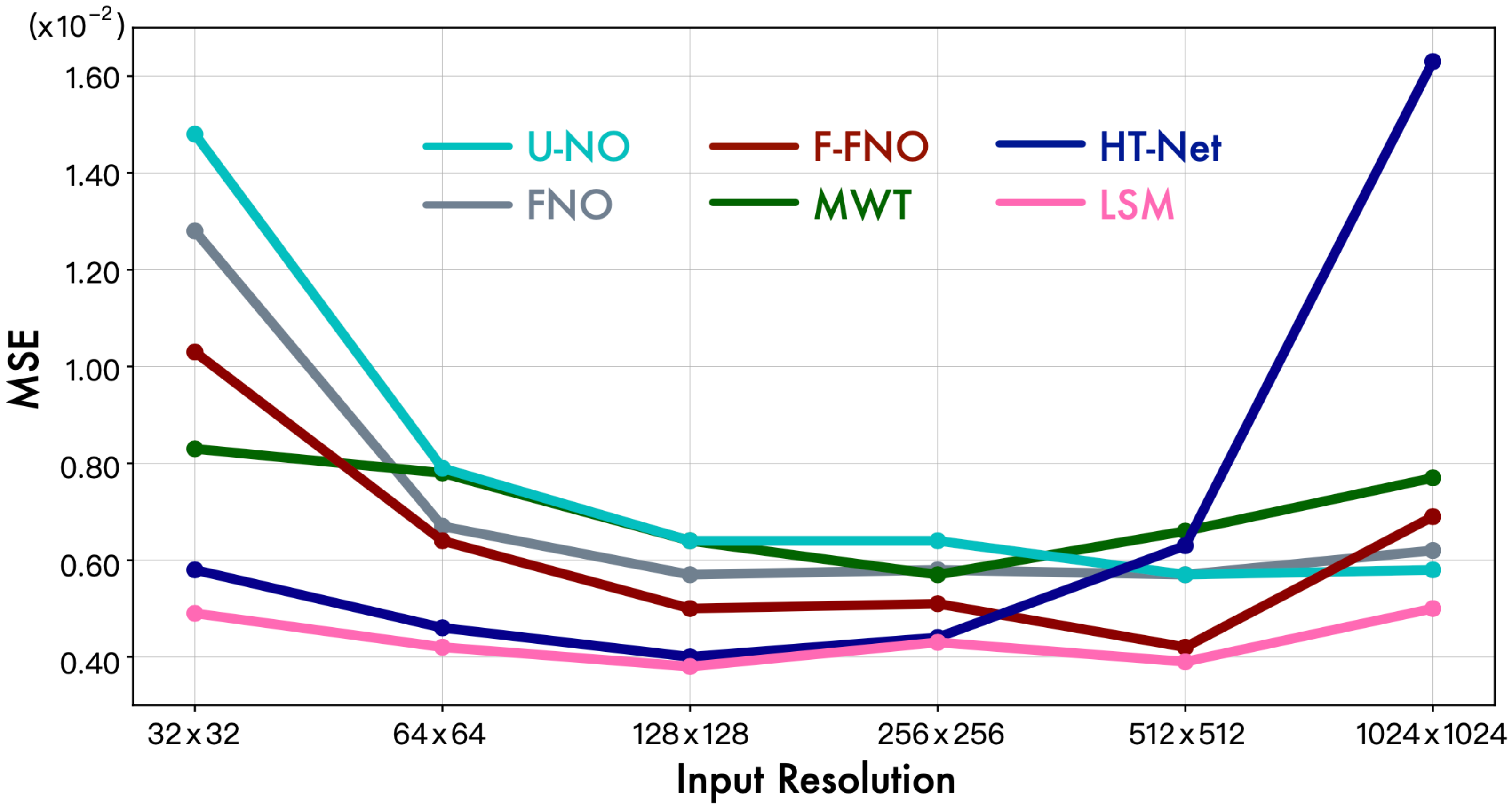


(a) Darcy Benchmark

(b) Latent Process on Navier-Stokes

LSM can precisely capture the complex mapping and latent process from high-dimensional coordinate space.

Performance under various resolutions in Darcy benchmark

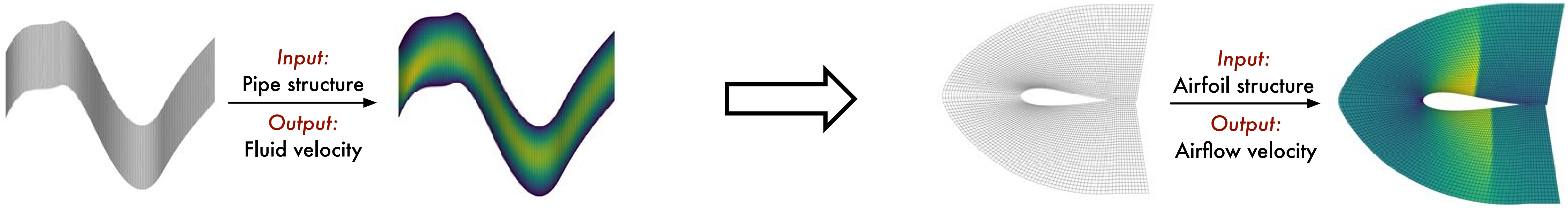


LSM presents a stable performance w.r.t. different inputs and consistently surpasses other baselines.

Transferability

Finetune the pipe-pretrained model into airfoil with limited data

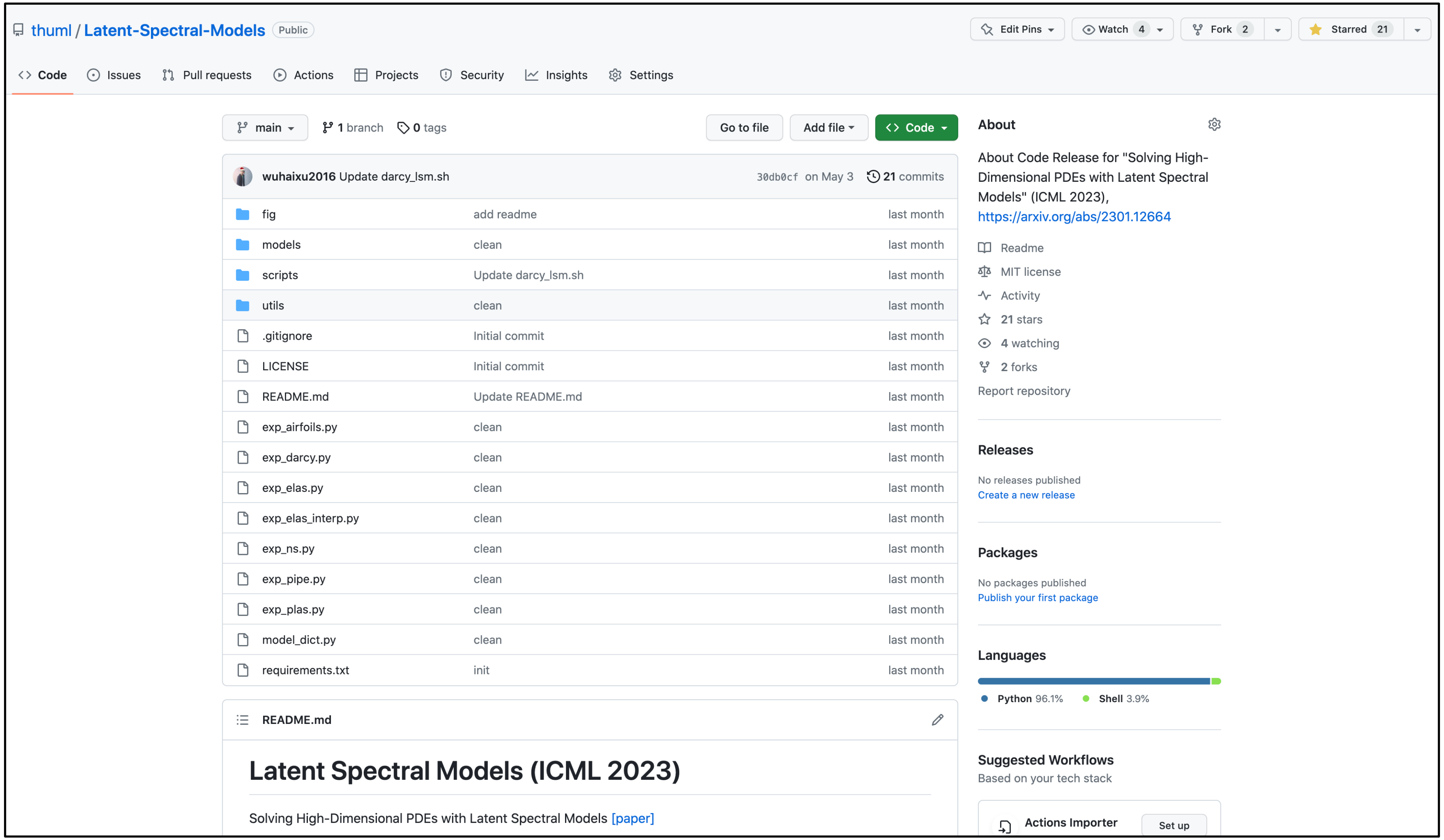
(same PDE equation but different boundary conditions)



MSE ($\times 10^{-2}$)	20% Airfoil Data	40% Airfoil Data	60% Airfoil Data	80% Airfoil Data	100% Airfoil Data
U-Net (2015)	1.88→1.93 (-2.7%)	1.38→1.14 (+17.3%)	0.96→0.90 (+6.3%)	0.85→0.81 (+4.7%)	0.79→0.77 (+2.5%)
U-NO (2022)	6.30→1.72	2.39→1.73	1.10→1.00 (+9.1%)	0.86→0.82 (+4.7%)	0.78→0.82 (-5.1%)
HT-Net (2022)	1.73→1.43 (+17.3%)	1.08→0.82 (**+24.1%**)	0.75→0.69 (+8.0%)	0.70→0.65 (+7.1%)	0.65→0.61 (+6.2%)
LSM	**1.66→1.31** (**+21.1%**)	**0.91→0.75** (+17.6%)	**0.69→0.61** (**+11.6%**)	**0.63→0.58** (**+7.9%**)	**0.59→0.55** (**+6.8%**)

LSM shows good Transferability between different conditions.

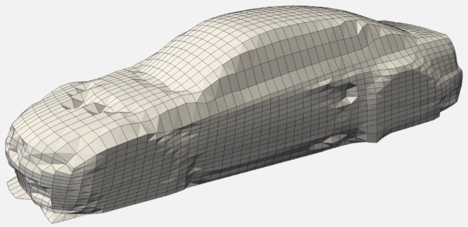
Open Source



Code is available at https://github.com/thuml/Latent-Spectral-Models

A Roadmap to Practical Neural PDE Solvers

Industrial simulation with CAE



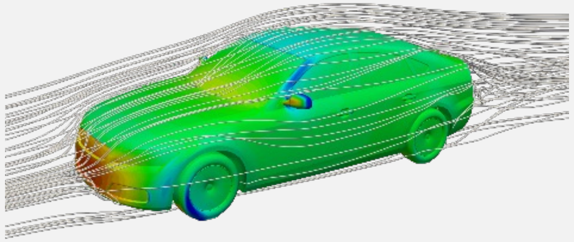
Varied Geometries

Ansys

DASSAULT SYSTEMES

SIEMENS

HEXAGON



Physical Simulation

Neural PDE Solver (Our work)

Q1: High-dimensional Mapping Approximation

Q2: Large-scale Irregular Meshes

Q3: Generalization among varied PDEs

Deep Models → Latent Spectral Models → **Transolver** → Unisolver → ...

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

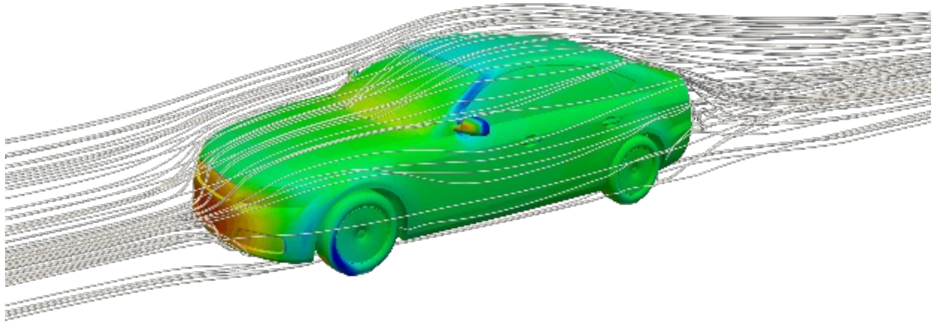

Huakun Luo

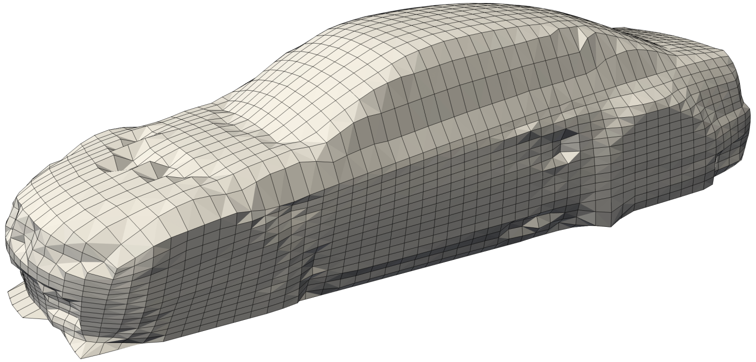

Haowen Wang


Jianmin Wang

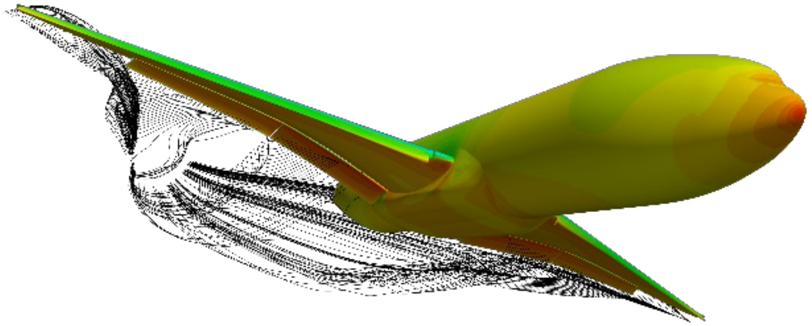

Mingsheng Long

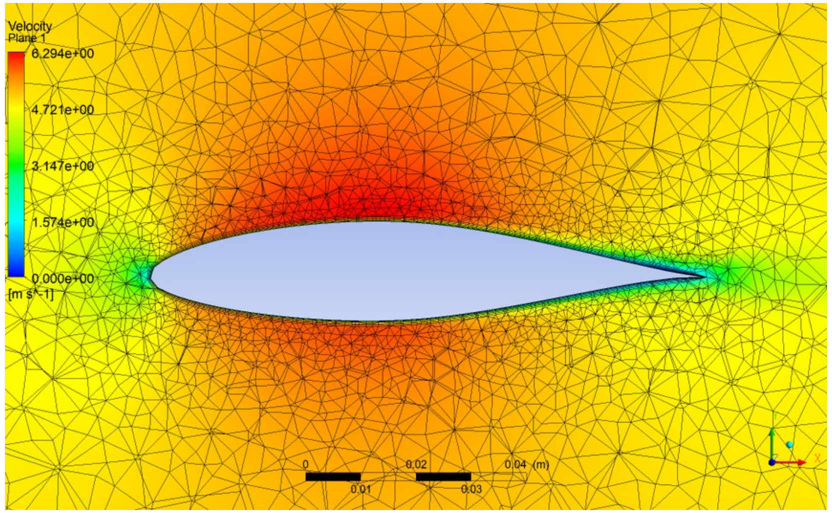
Solving PDEs: Discretization





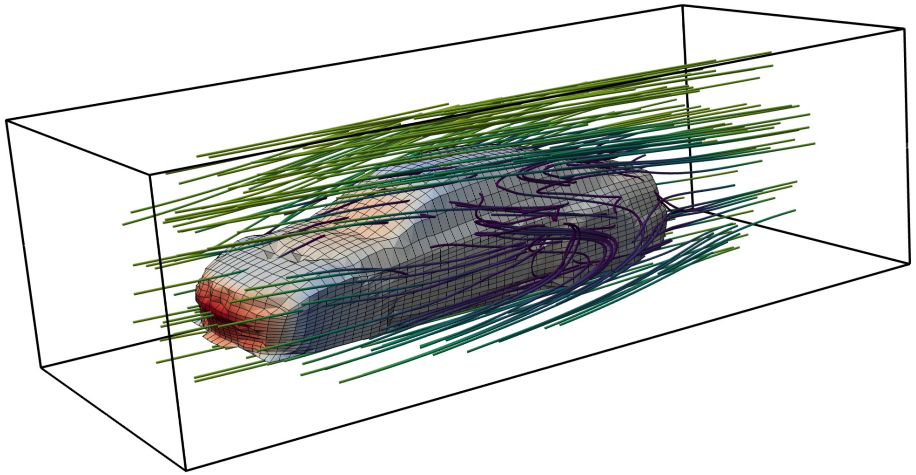
Car





Airplane

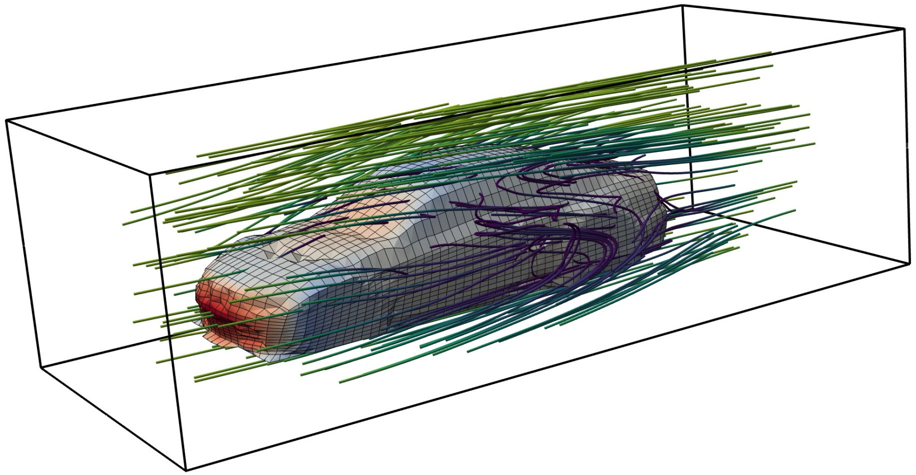
Challenges in Practical Industrial Design



Example: Estimate the drag coefficient of a given shape:

Surrounding Wind & Surface Pressure

Challenges in Practical Industrial Design

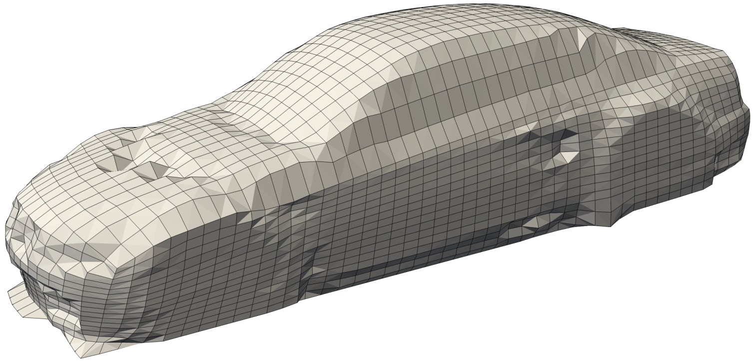


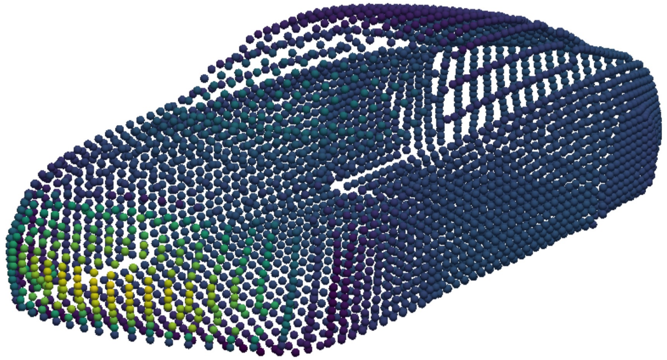
Example: Estimate the drag coefficient of a given shape:

Surrounding Wind & Surface Pressure

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

Previous Work: Geometric Deep Learning





(1) Mesh

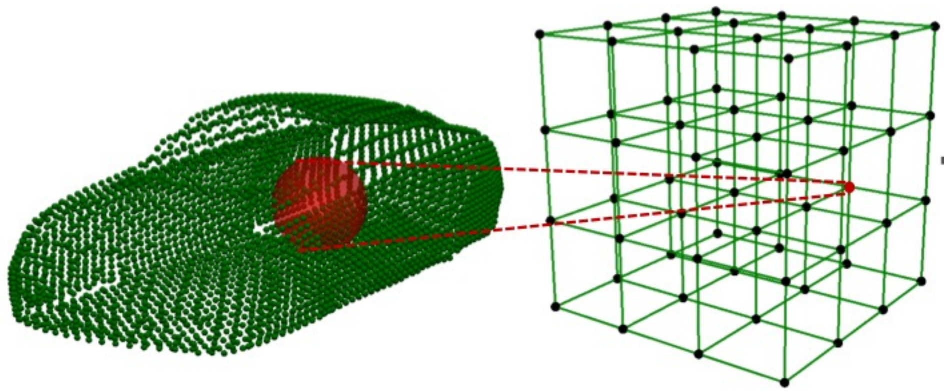
GraphSAGE, MeshGraphNet, etc

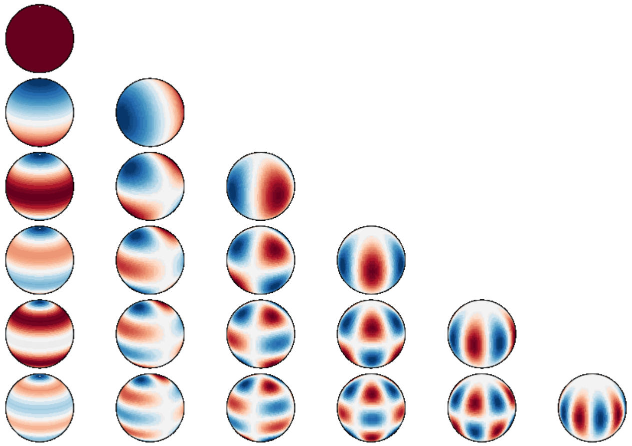
(2) Point Cloud

PointNet, Point Transformer, etc

Excels in geometry modeling but fail in physics learning

Previous Work: Geometry-General Neural Operators





(1) GNN as Operators

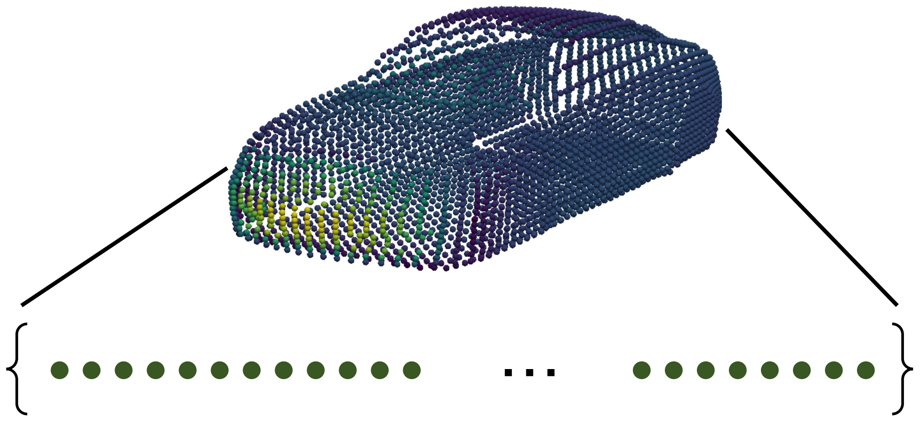
GNO, GINO, etc

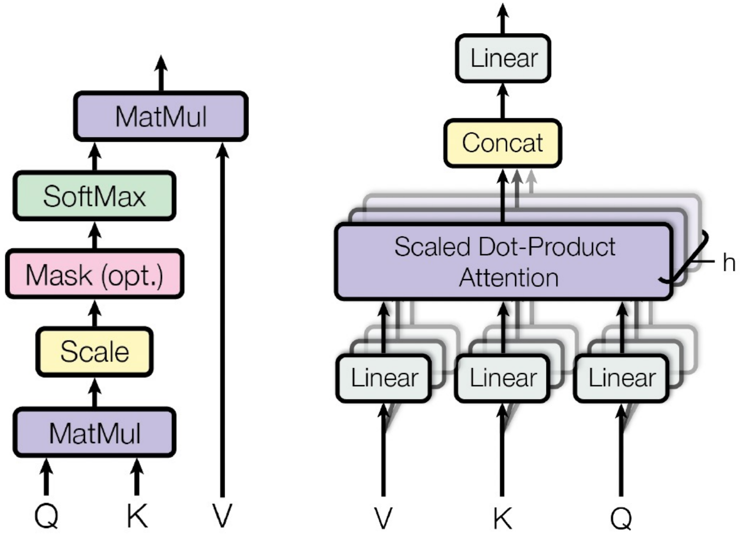
(2) FNO-Variants

geoFNO, SFNO, etc

Only focus on local physics or limited to periodic boundary

Transformer-based PDE Solvers



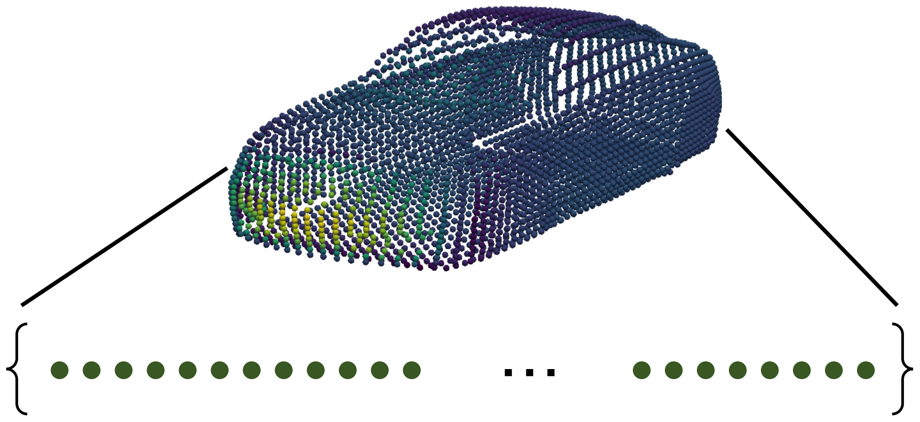


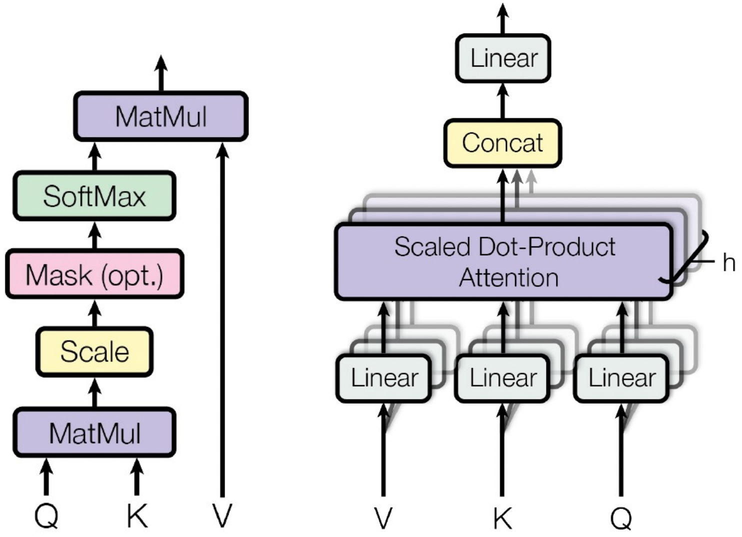
(1) Geometries as point sequences (2) Attention as Monte Carlo Integral

OFormer, Galerkin Transformer, etc

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

Transformer-based PDE Solvers





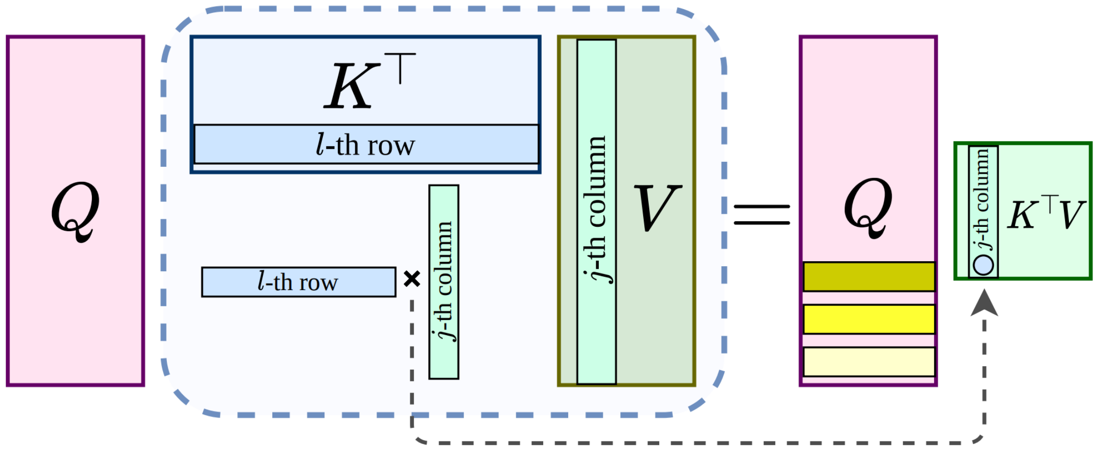
(1) Geometries as point sequences (2) Attention as Monte Carlo Integral

OFormer, Galerkin Transformer, etc

How to efficiently capture physical correlations underlying discretized meshes

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

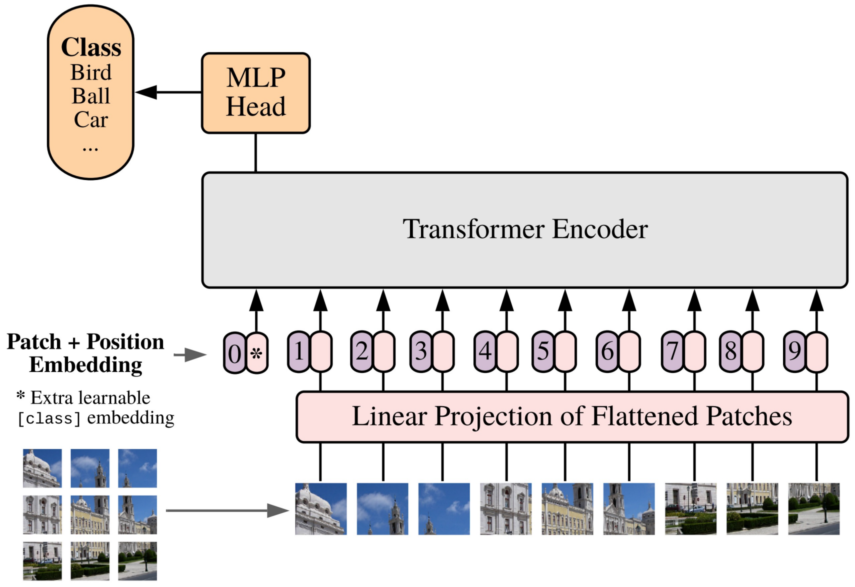
Related Work



(1) Linear Transformers

1. Less informative attention

2. Individual points is insufficient for physics learning

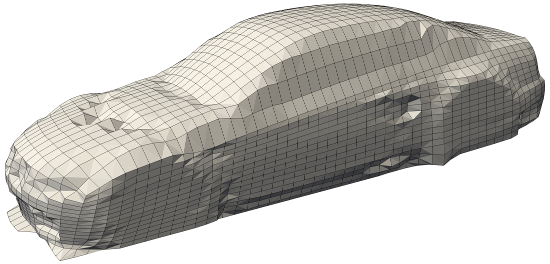


(2) Vision Transformer

Augment features with patch ✓

Not applicable to irregular meshes

A foundational Idea of Transolver



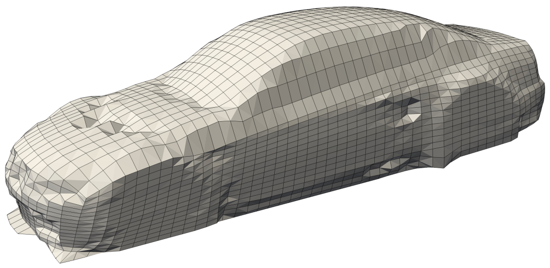
Discretized Domain

Previous Work

Being “trapped” to superficial and unwieldy meshes

Difficulties in Complexity, Geometry, Physics

A foundational Idea of Transolver

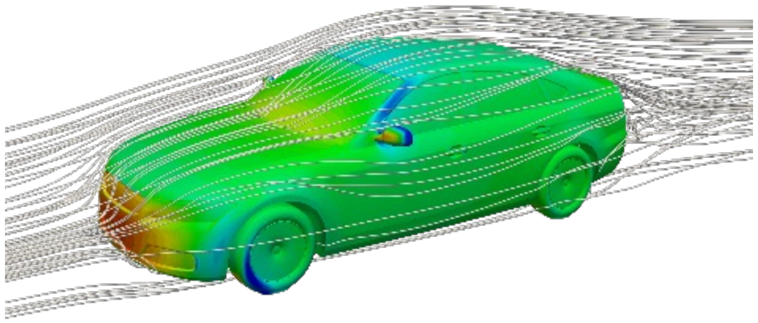


Discretized Domain

Previous Work

Being “trapped” to superficial and unwieldy meshes

Difficulties in Complexity, Geometry, Physics



Physics Domain

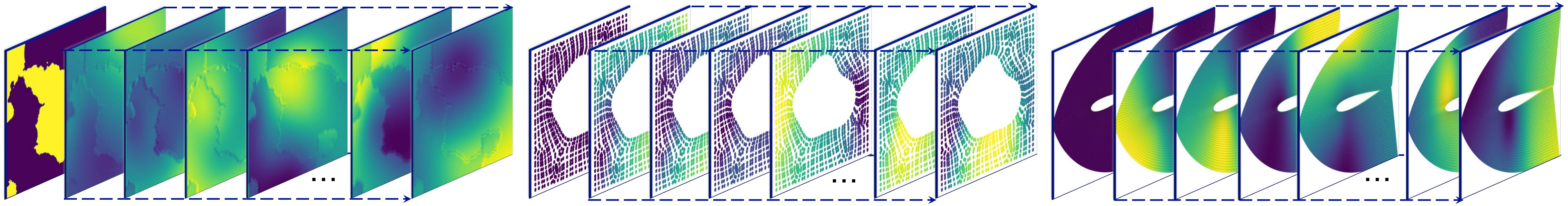
Transolver

Learning **intrinsic physical states** under

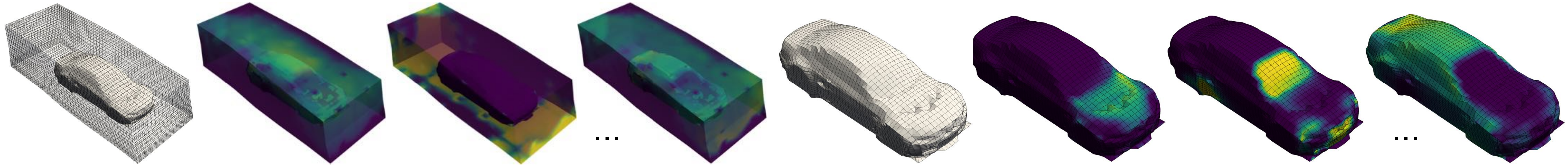
complex and large-scale geometrics

Better Complexity, Geometry, Physics Modeling

Learning Physical States



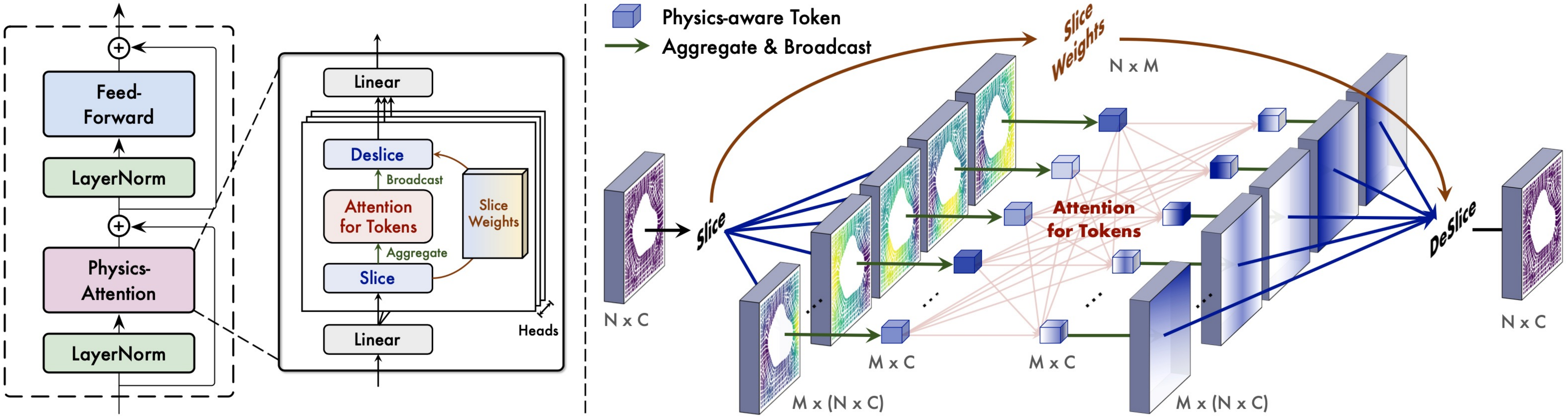
(a) Slices for Darcy, 2D Regular Grid (b) Slices for Elasticity, 2D Point Cloud (c) Slices for Airfoil, 2D Mesh



(d) Slices for Shape-Net Car Surrounding Velocity, 3D Volumes (e) Slices for Shape-Net Car Surface Pressure, 3D Mesh

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

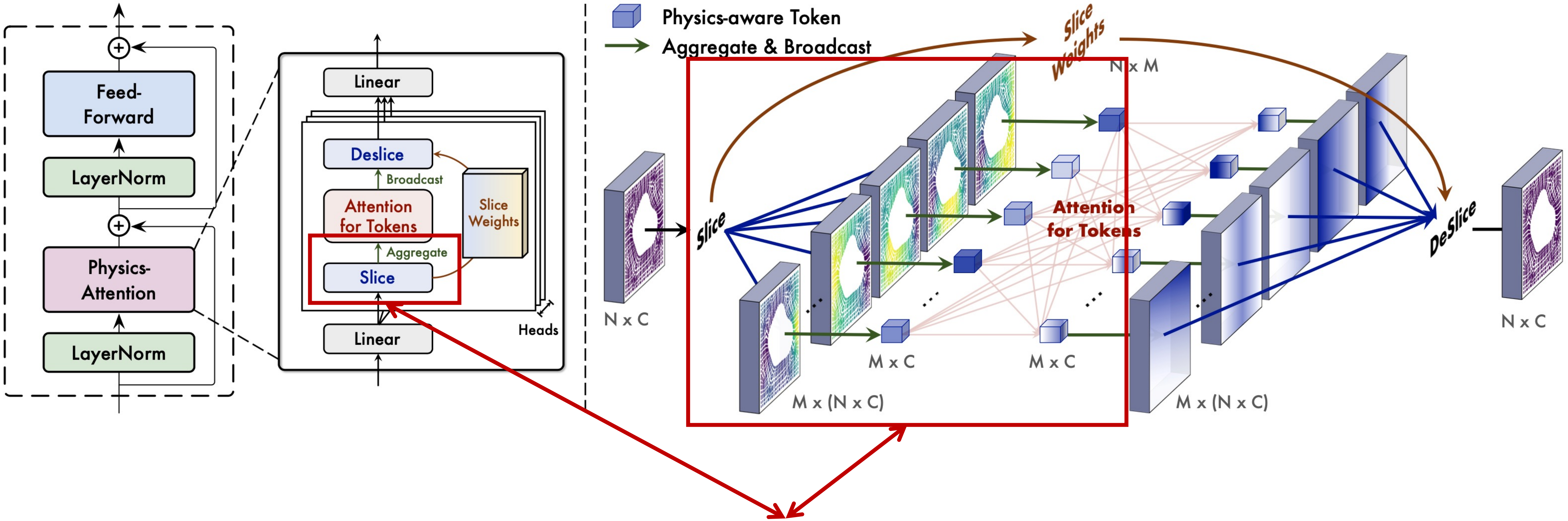
Overview of Transolver



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

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

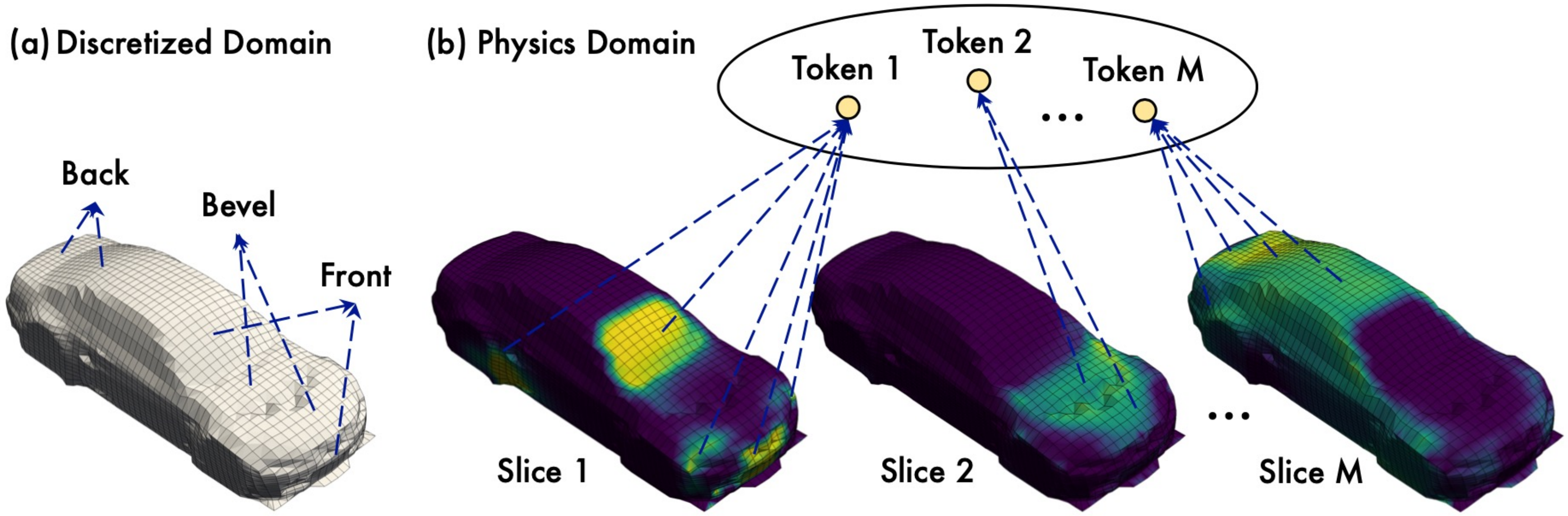
Overview of Transolver



① Mesh → physics

To obtain physics-aware tokens

Mesh → physics



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

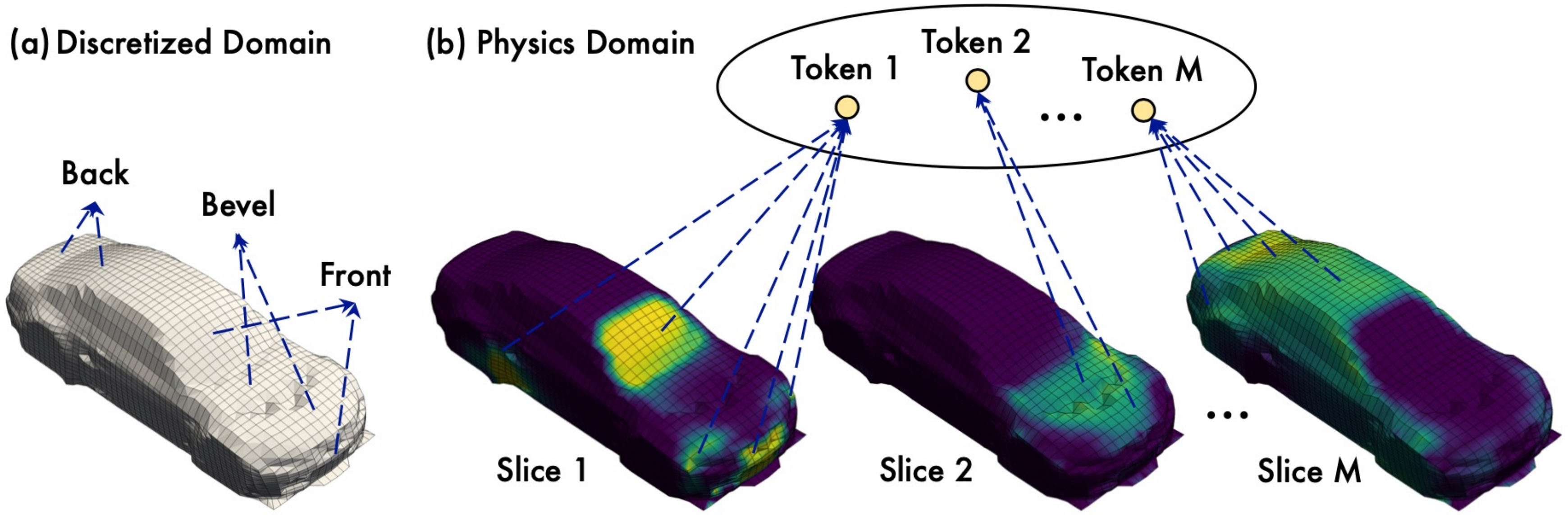
$$\{\mathbf{w}_i\}_{i=1}^N = \left\{\underline{\mathrm{Softmax}}\left(\mathrm{Project}\left(\mathbf{x}_i\right)\right)\right\}_{i=1}^N$$

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

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

Softmax for low-entropy slices

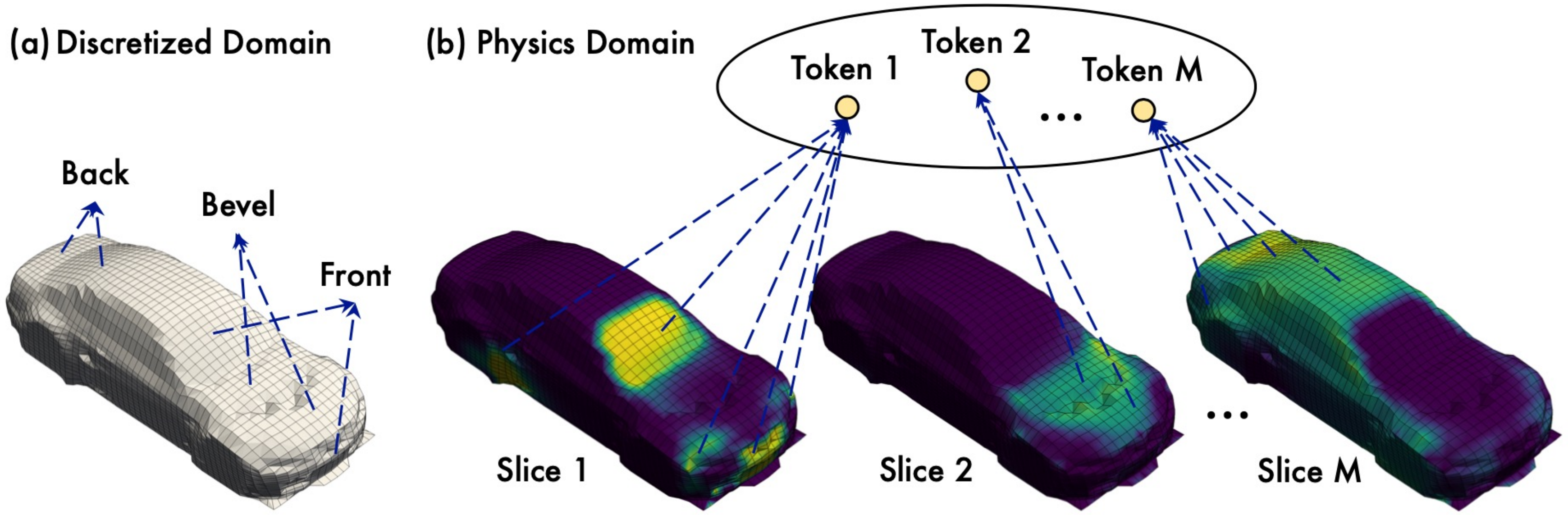
Mesh → physics



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

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

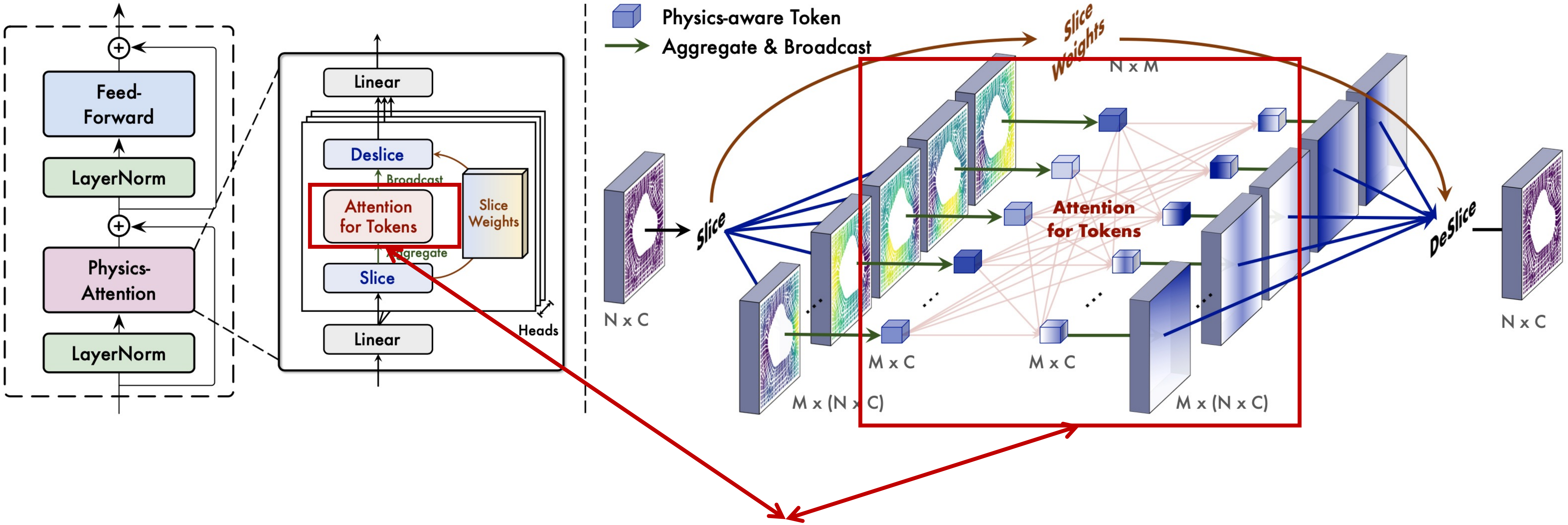
Mesh → physics



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

   Ascribe physically similar but spatially distant points to the same slice

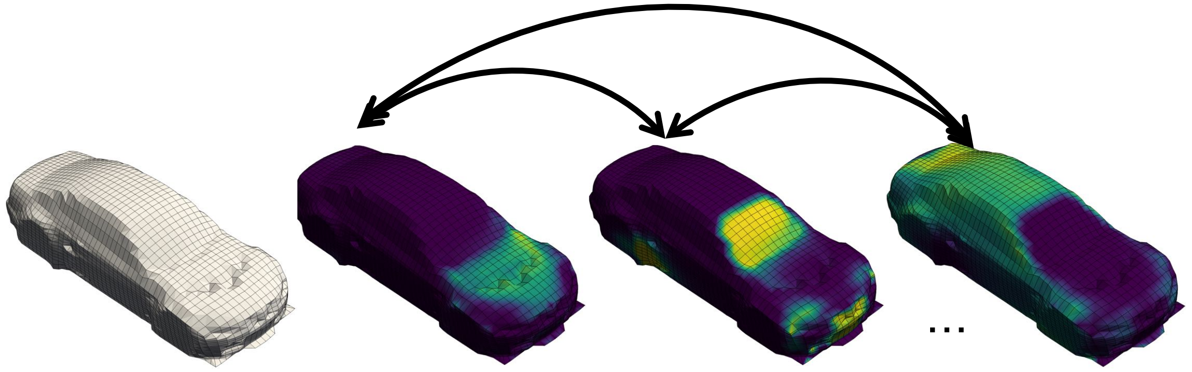
Overview of Transolver



② Attention among physics tokens

Approximate Integral to solve PDEs

Attention among physics tokens

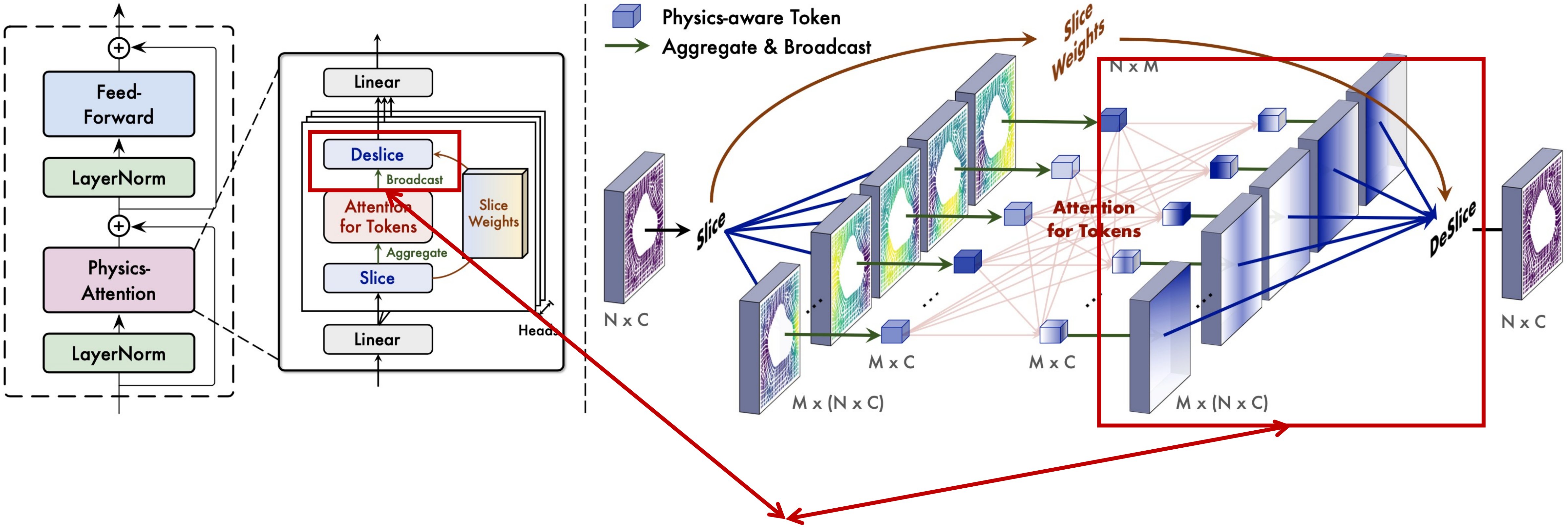


$$\mathbf{q}, \mathbf{k}, \mathbf{v} = \mathrm{Linear}(\underline{\mathbf{z}}), \quad \mathbf{z}' = \mathrm{Softmax}\left(\frac{\mathbf{q}\mathbf{k}^{\mathsf{T}}}{\sqrt{C}}\right)\mathbf{v}$$

Canonical attention among physics tokens

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

Overview of Transolver



③ Physics → Mesh

Project physics information back to mesh

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

Theoretical Understanding of Transolver

1. Corollary of *Attention is a learnable integral*

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

Is Physics-Attention still an input domain integral?

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

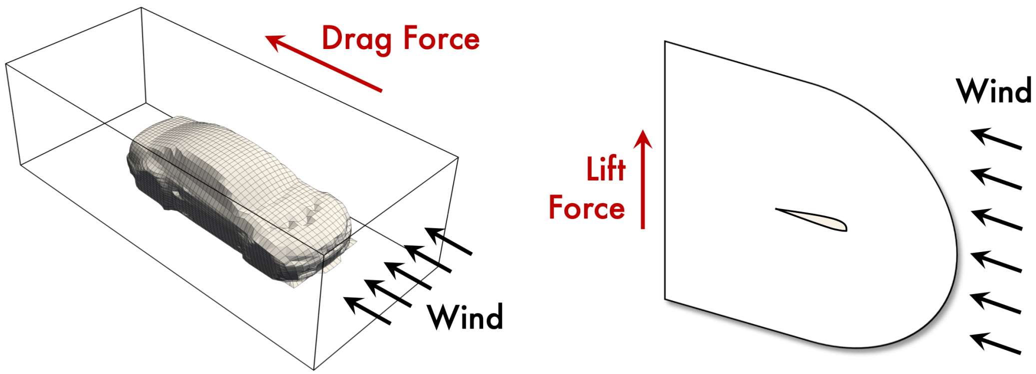
Theoretical Understanding of Transolver

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

All the designs in Transolver can be directly derived.

Experiments

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



(a) Shape-Net Car (b) AirfRANS

Six standard benchmarks, two practical design tasks

More than 20 baselines

Standard PDE-Solving Benchmarks

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

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

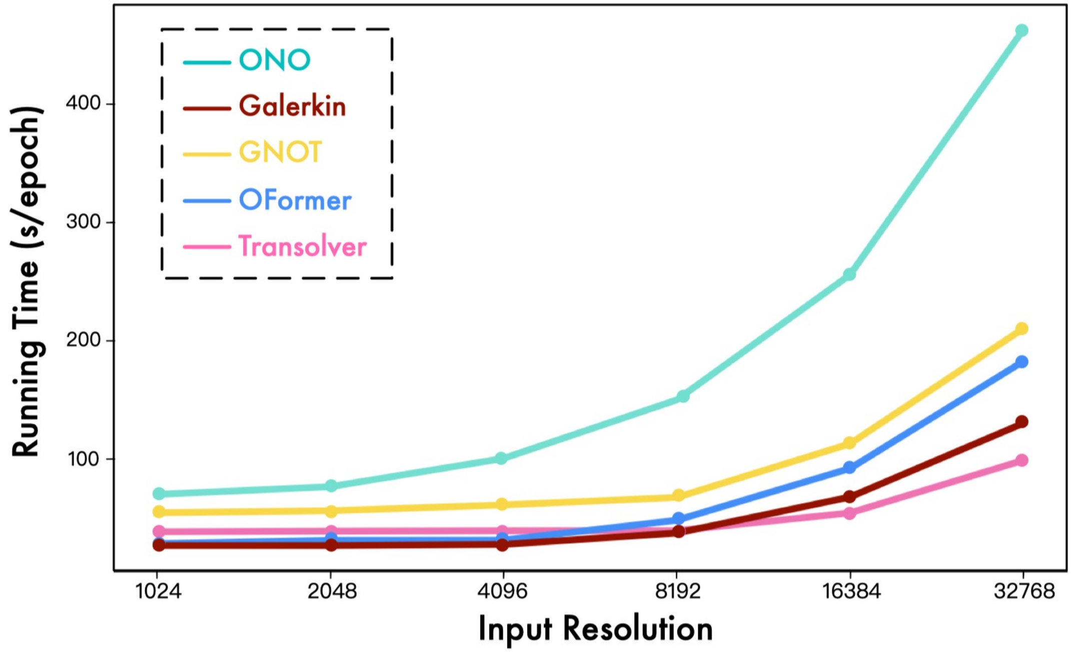
Practical Design Tasks

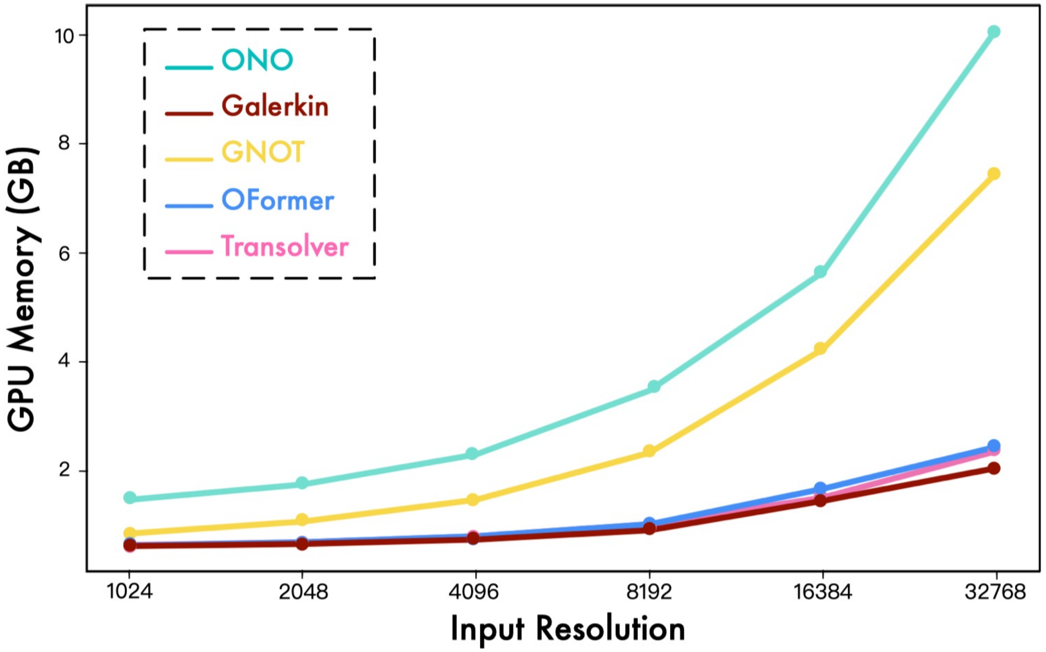
Model*	Shape-Net Car				AirfRANS			
	Volume ↓	Surf ↓	C_D ↓	ρ_D ↑	Volume ↓	Surf ↓	C_L ↓	ρ_L ↑
Simple MLP	0.0512	0.1304	0.0307	0.9496	0.0081	0.0200	0.2108	0.9932
GraphSAGE (Hamilton et al., 2017)	0.0461	0.1050	0.0270	0.9695	0.0087	0.0184	0.1476	0.9964
PointNet (Qi et al., 2017)	0.0494	0.1104	0.0298	0.9583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

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

Transolver performs best in both physics and design-oriented metrics

Efficiency

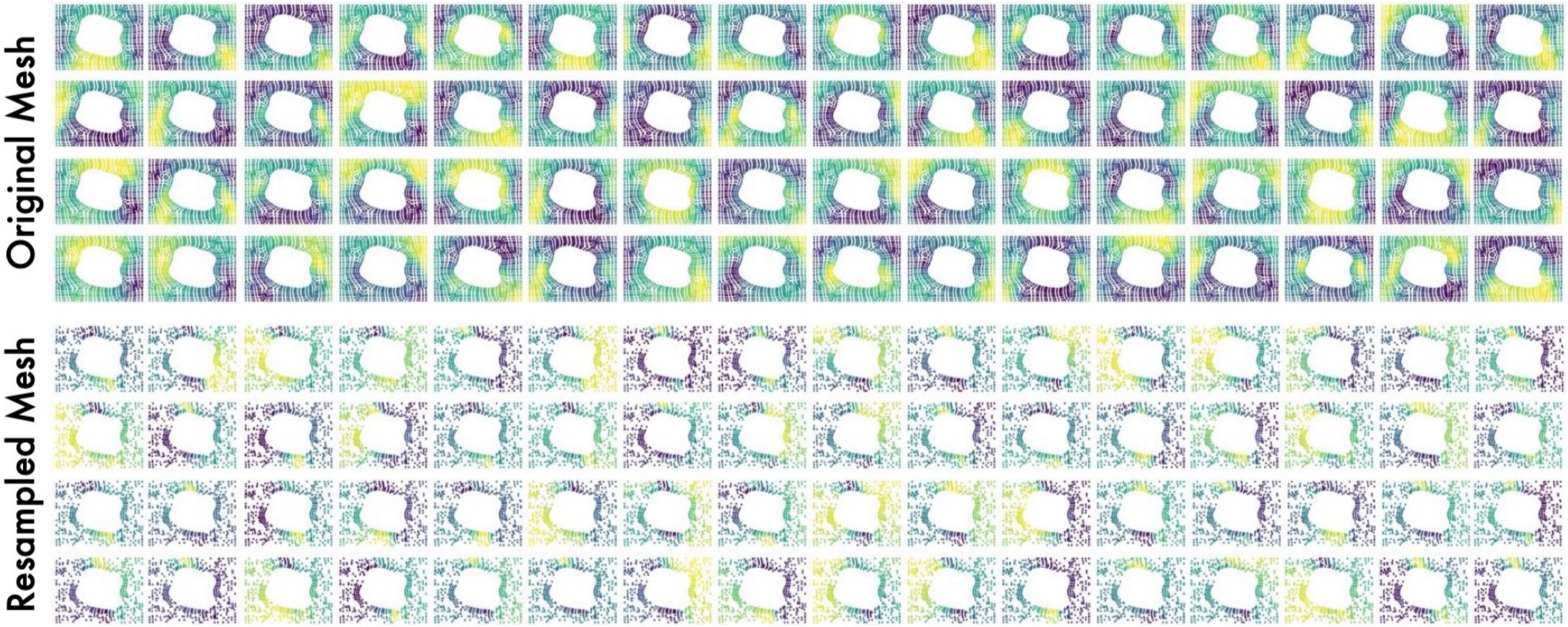




Favorable efficiency and performance balance

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

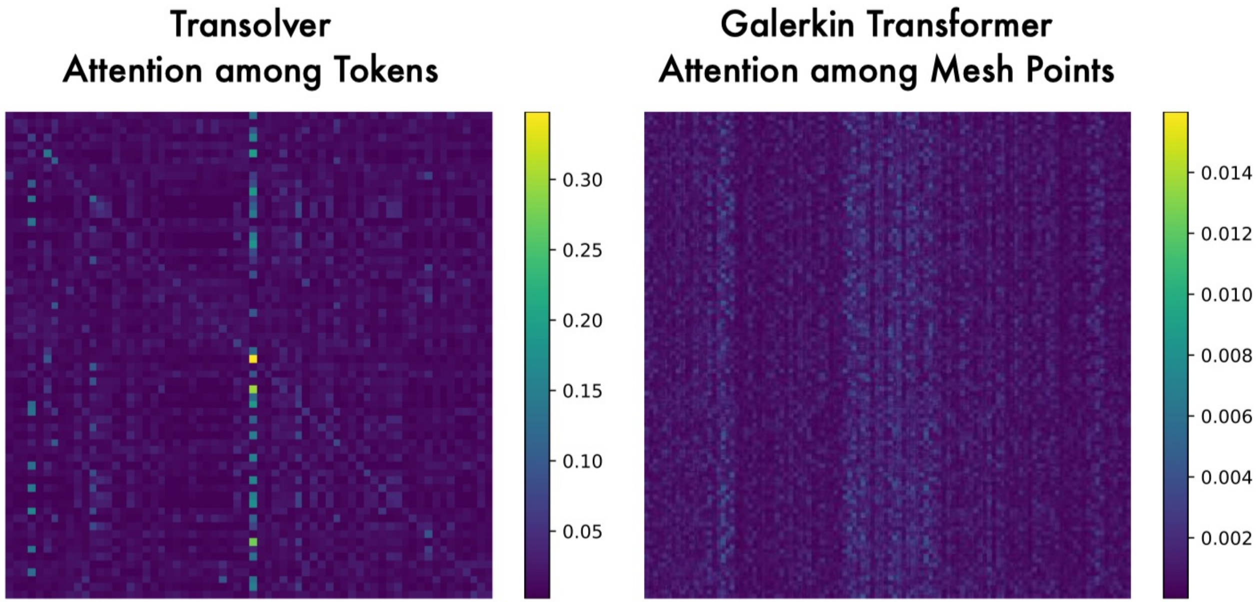
Physics-Attention Visualization



Slice visualization on Elasticity

Transolver is mesh-free, precisely captures states **even on broken meshes**

Physics-Attention Visualization

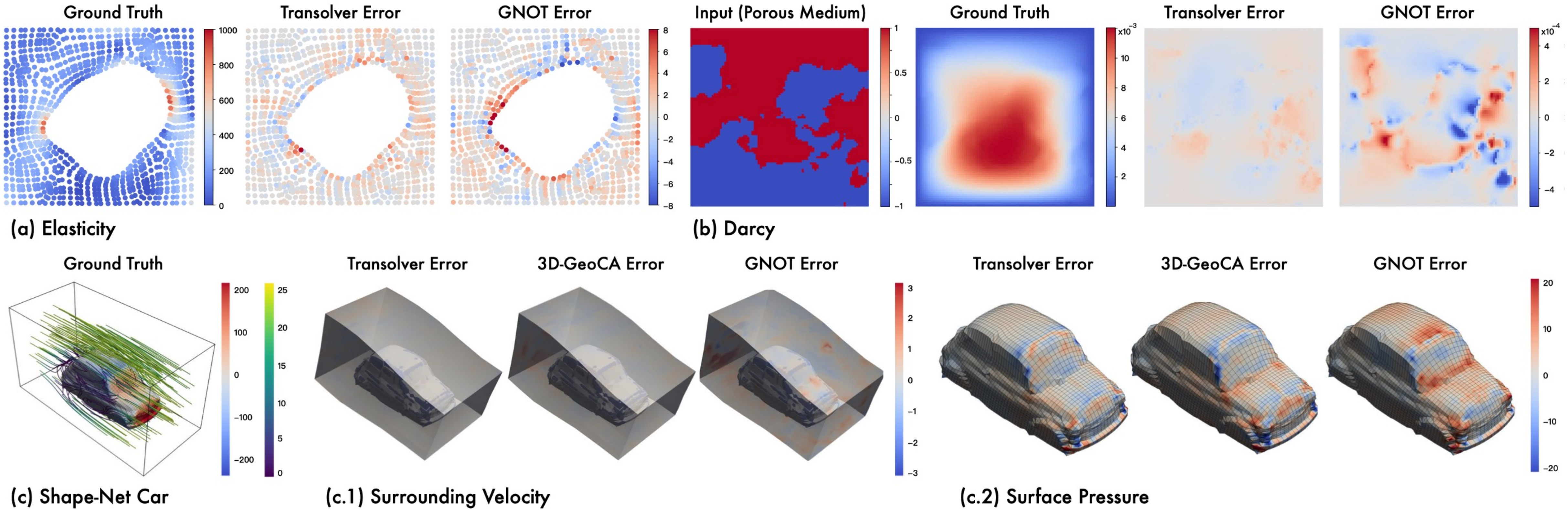


Kullback–Leibler (KL) divergence between attention weights and uniform distribution

Benchmarks	Galerkin (Cao, 2021)	Transolver (Ours)
Elasticity (972 mesh points)	0.3803	**1.7795**
Darcy (7,225 mesh points)	0.2739	**1.8274**

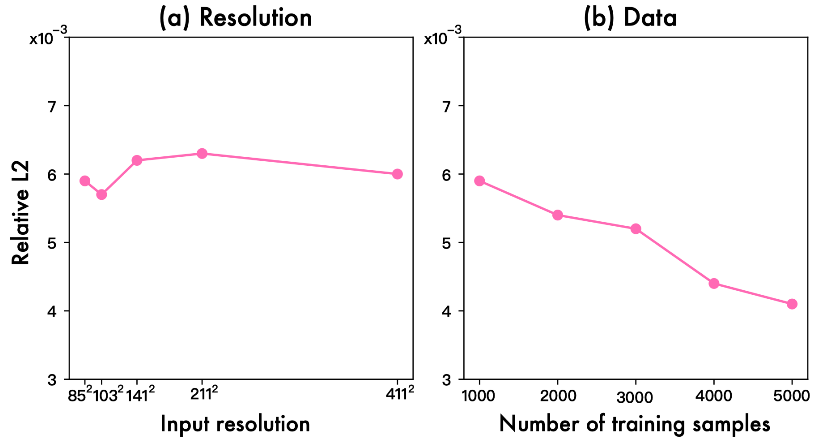
Physics-Attention can learn **more informative physical correlations**

Showcases

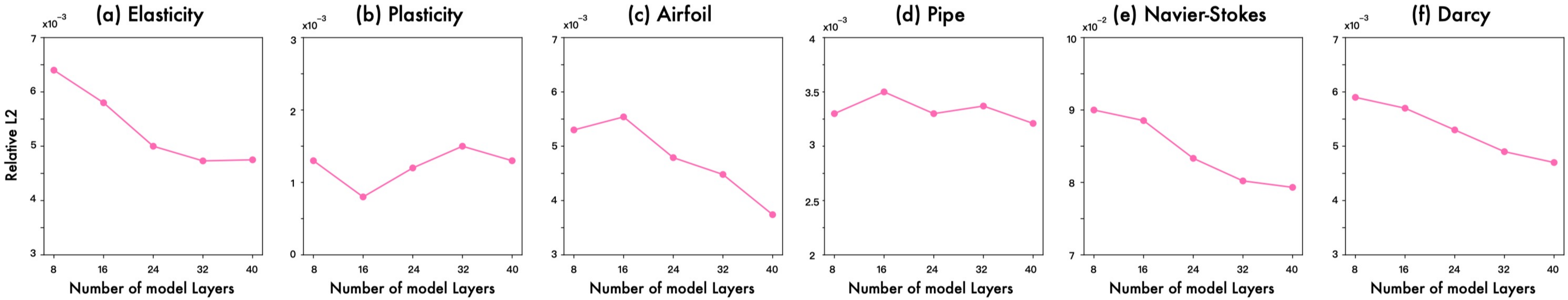


Transolver excels in solving **multiphysics PDEs on hybrid geometrics**

Pursuing PDE Foundation Models: **Scalability**

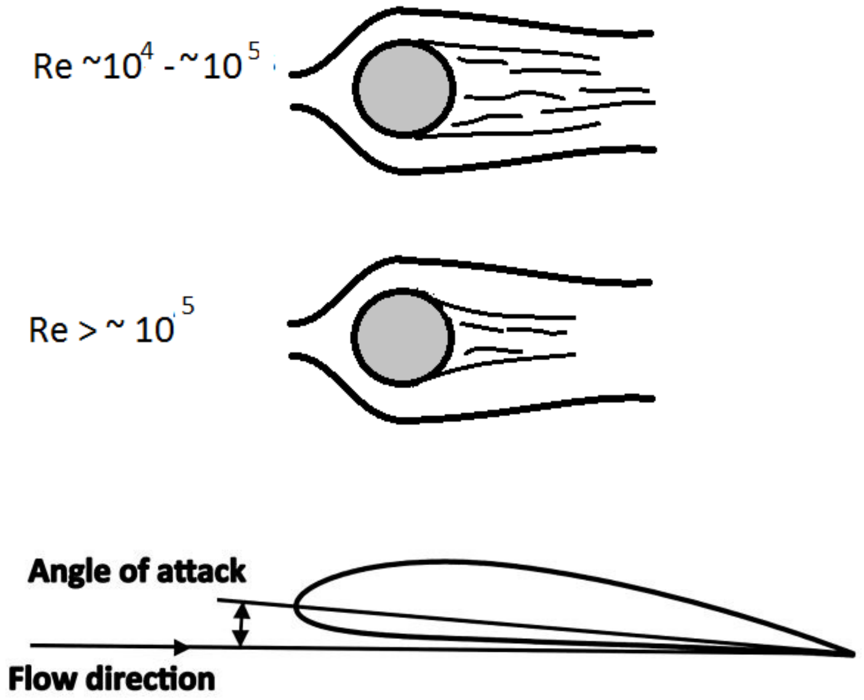


1. **Resolution:** Consistent performance at varied scales
2. **Data:** Benefiting from larger training data
3. **Parameter:** Benefiting from more parameters



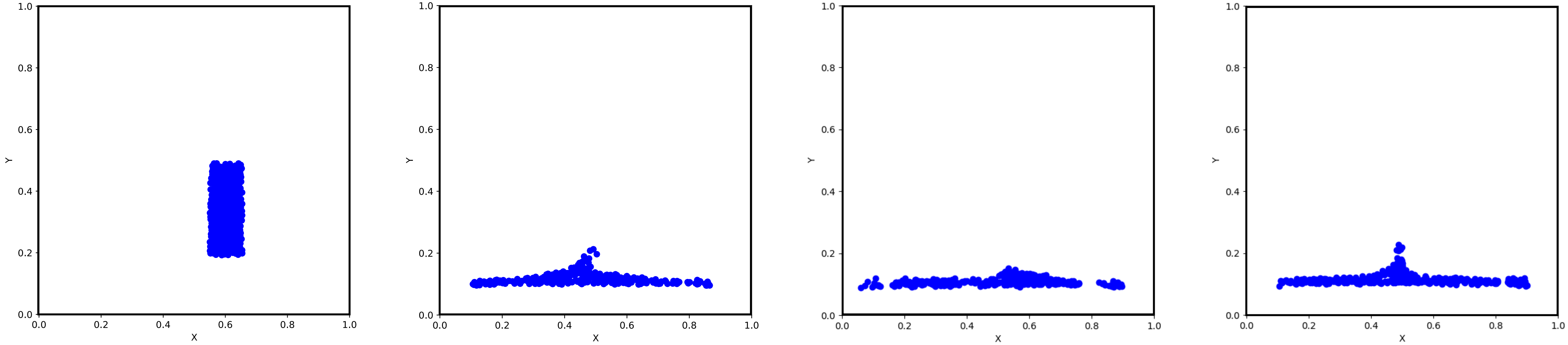
Pursuing PDE Foundation Models: **Generalization**

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



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

Pursuing PDE Foundation Models: **Versatile**



Initial State   Ground Truth (400th step)   GNN   GNN + Transolver

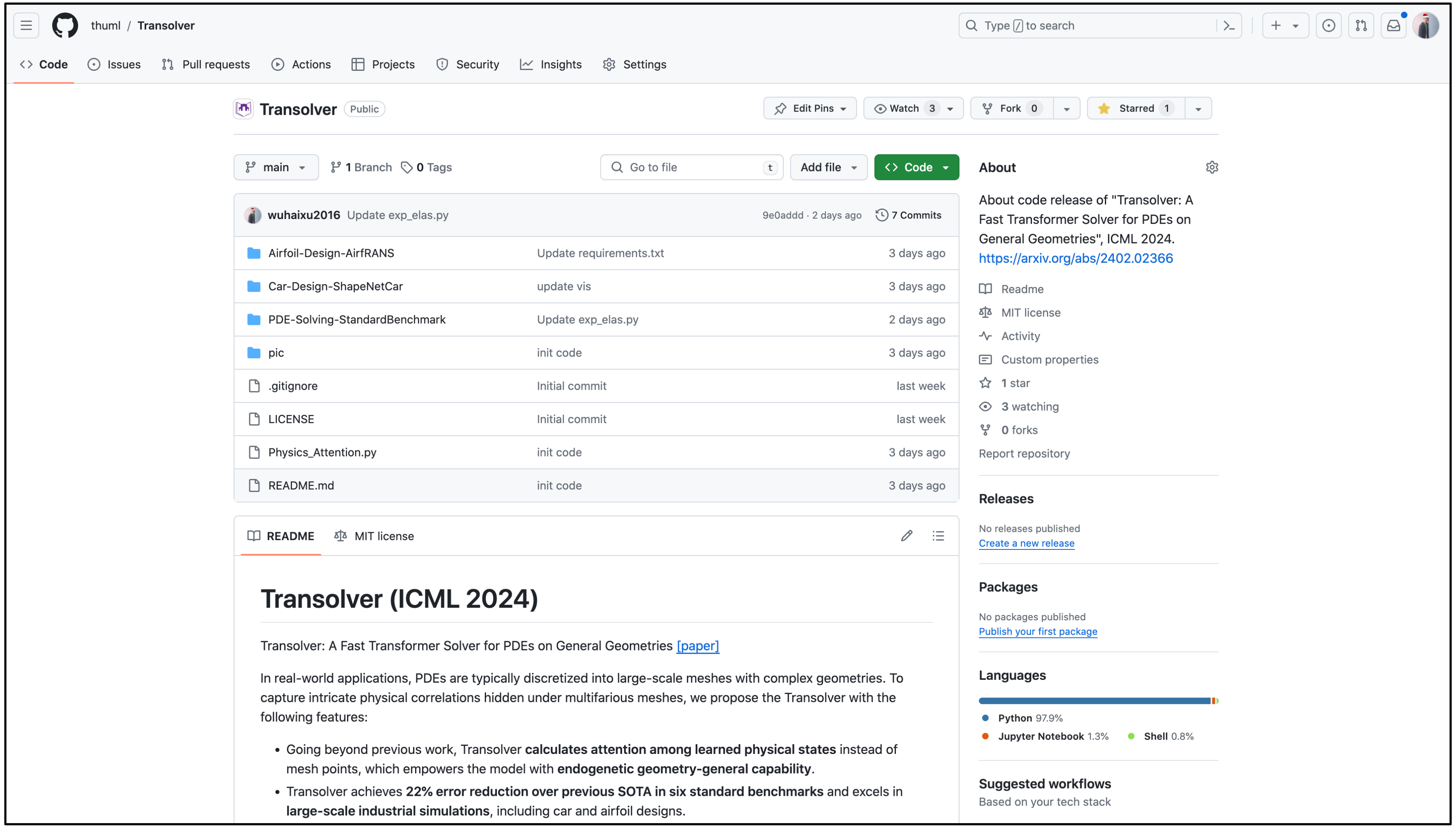
MODEL	MSE ↓
GNN (SANCHEZ-GONZALEZ ET AL., 2020)	0.0182
GNN + TRANSOLVER (OURS)	**0.0069**
RELATIVE PROMOTION	62.1%

Transolver can also be extended to

Lagrangian Settings

(Ever-changing geometrics)

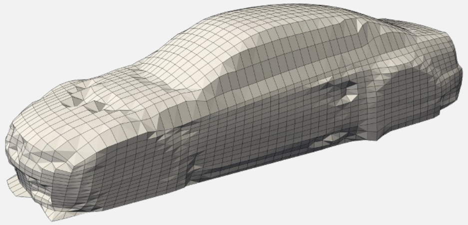
Open Source



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

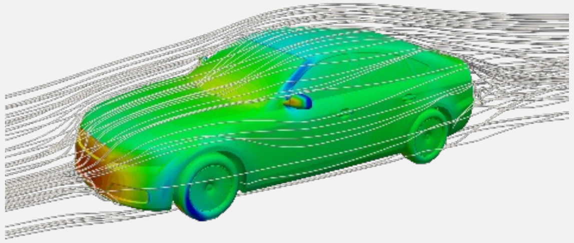
A Roadmap to Practical Neural PDE Solvers

Industrial simulation with CAE



Varied Geometries

Ansys DASSAULT SYSTEMES SIEMENS HEXAGON



Physical Simulation

Neural PDE Solver (Our work)

Q1: High-dimensional Mapping Approximation

Q2: Large-scale Irregular Meshes

Q3: Generalization among varied PDEs

Deep Models → Latent Spectral Models → Transolver → **Unisolver** → ...

Unisolver: PDE-Conditional Transformers Are Universal PDE Solvers

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


Hang Zhou

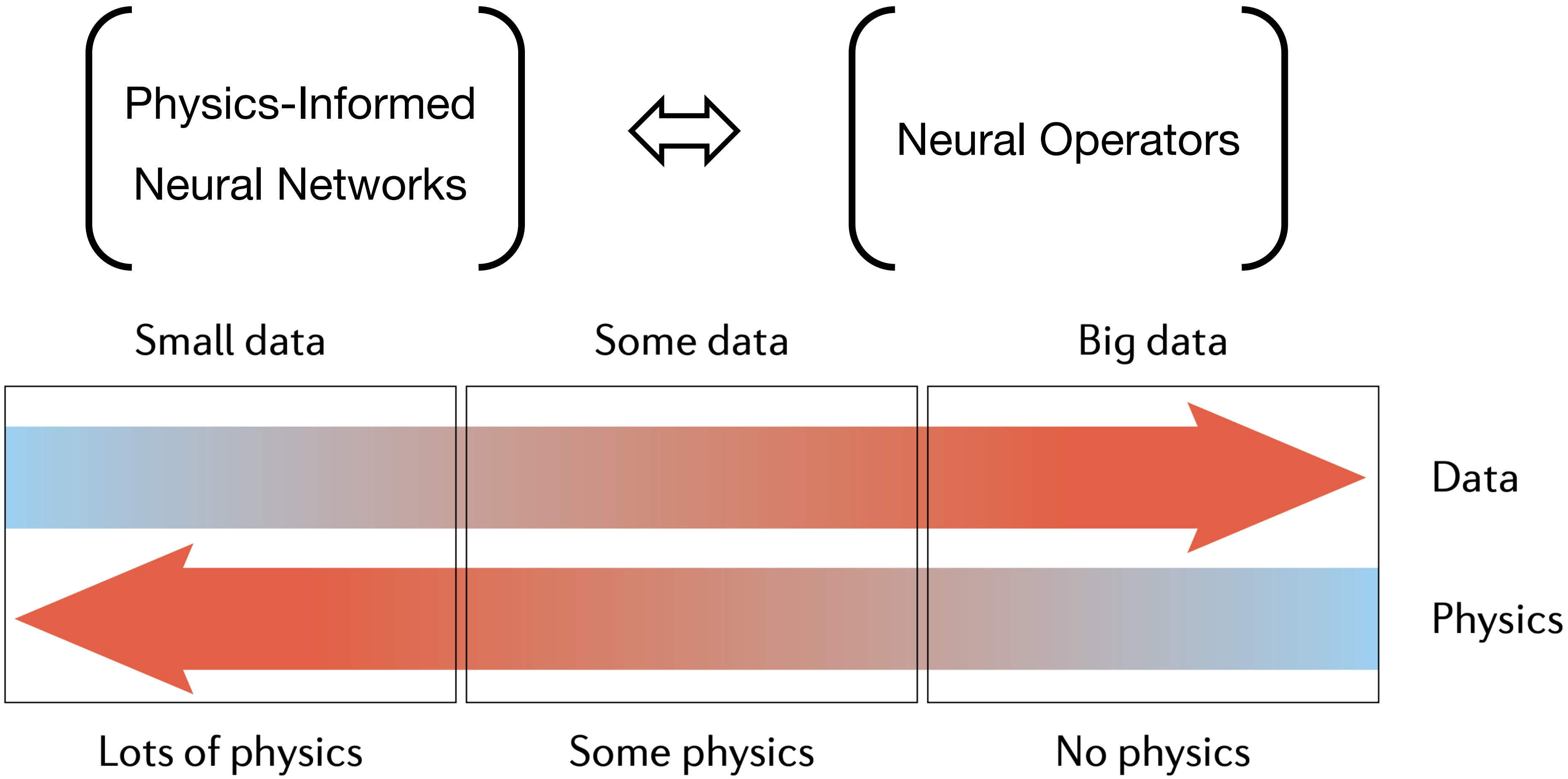
Yuezhou Ma


Haixu Wu


Haowen Wang


Mingsheng Long

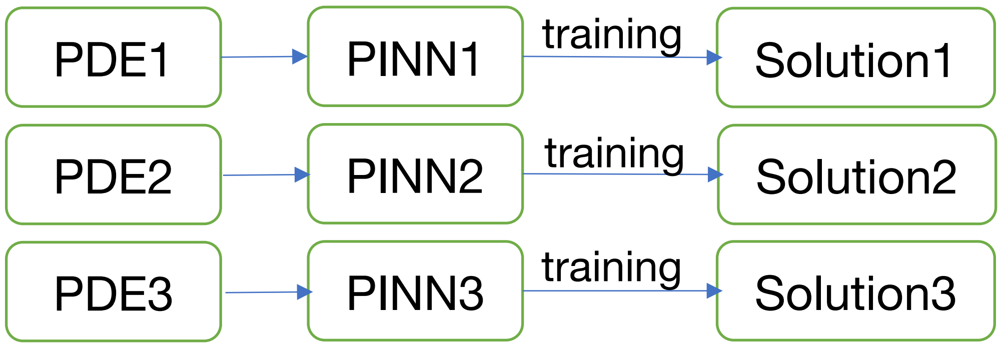
Machine Learning for PDEs



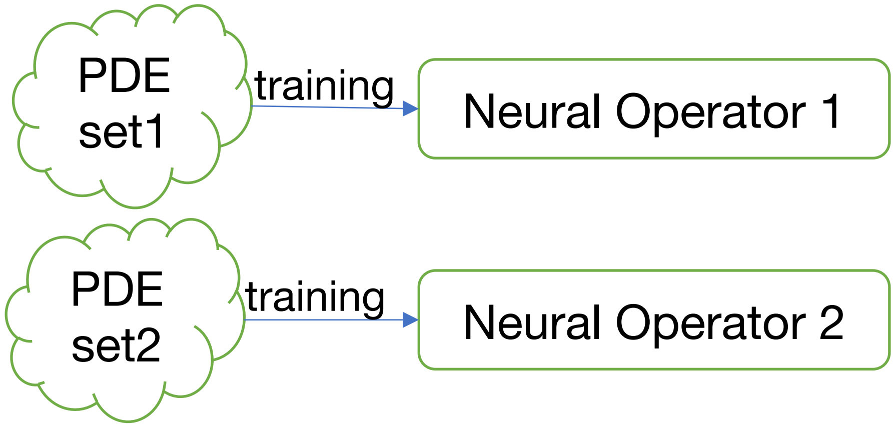
Karniadakis, G. et al. Physics-informed machine learning, ***Nature Review Physics*** 3, 422–440 (2021)

Generalizability of Neural PDE Solvers

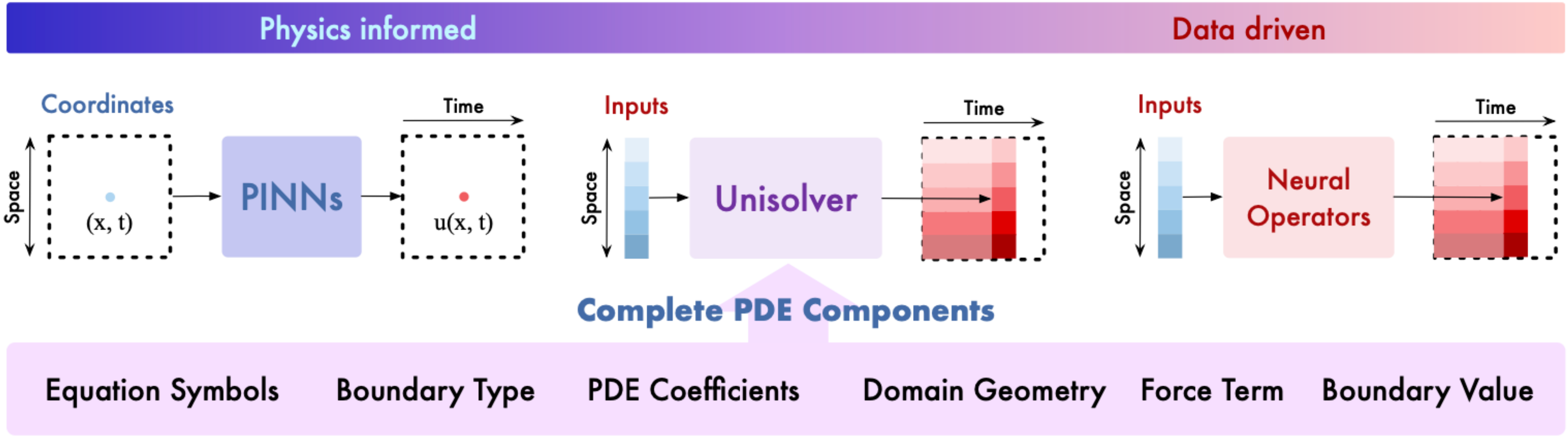
- PINNs
  1. Formalize the specific PDE equations as objective functions during training
  2. Struggle to generalize to unseen PDEs
  3. Re-training required to solve new PDEs



- Neural operators
  1. learning from pre-computed simulation data
  2. Cannot efficiently adapt to PDEs with varying components
  3. Computational costly and data-demanding



Unisolver: A Unification of Two Paradigms



In addition to simulated data, Unisolver also defines and utilizes **a complete set of PDE components**.

Complete PDE Components

Motivating example: vibrating string equation

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

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

Complete PDE Components

Motivating example: vibrating string equation

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

- The analytical solution of the above equations is:

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

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

Complete PDE Components

Categorization of PDE components

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

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

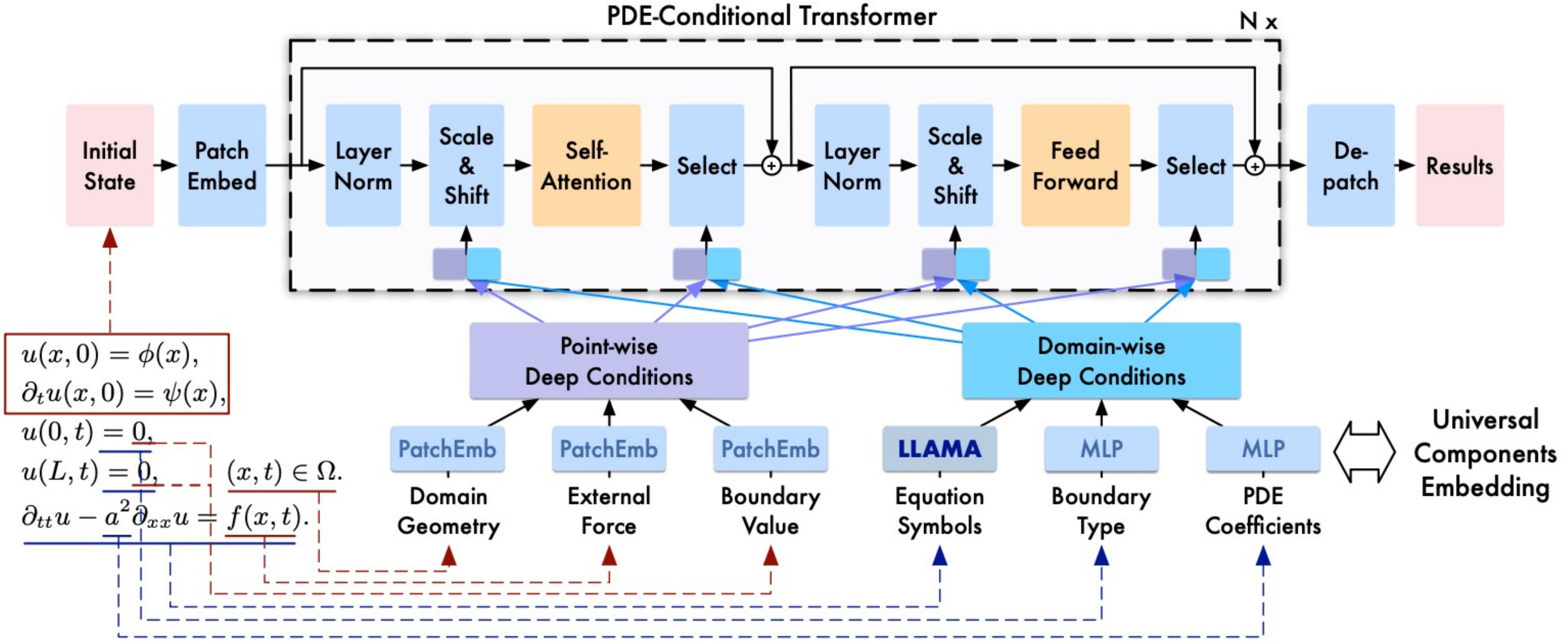
Universal Components Embedding

- ☐ Embedding of equation formulation
  - ✓ Utilizing a LLM to embed the symbolic expression of the PDE
  - ✓ The symbolic expression is represented by the LaTeX code

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

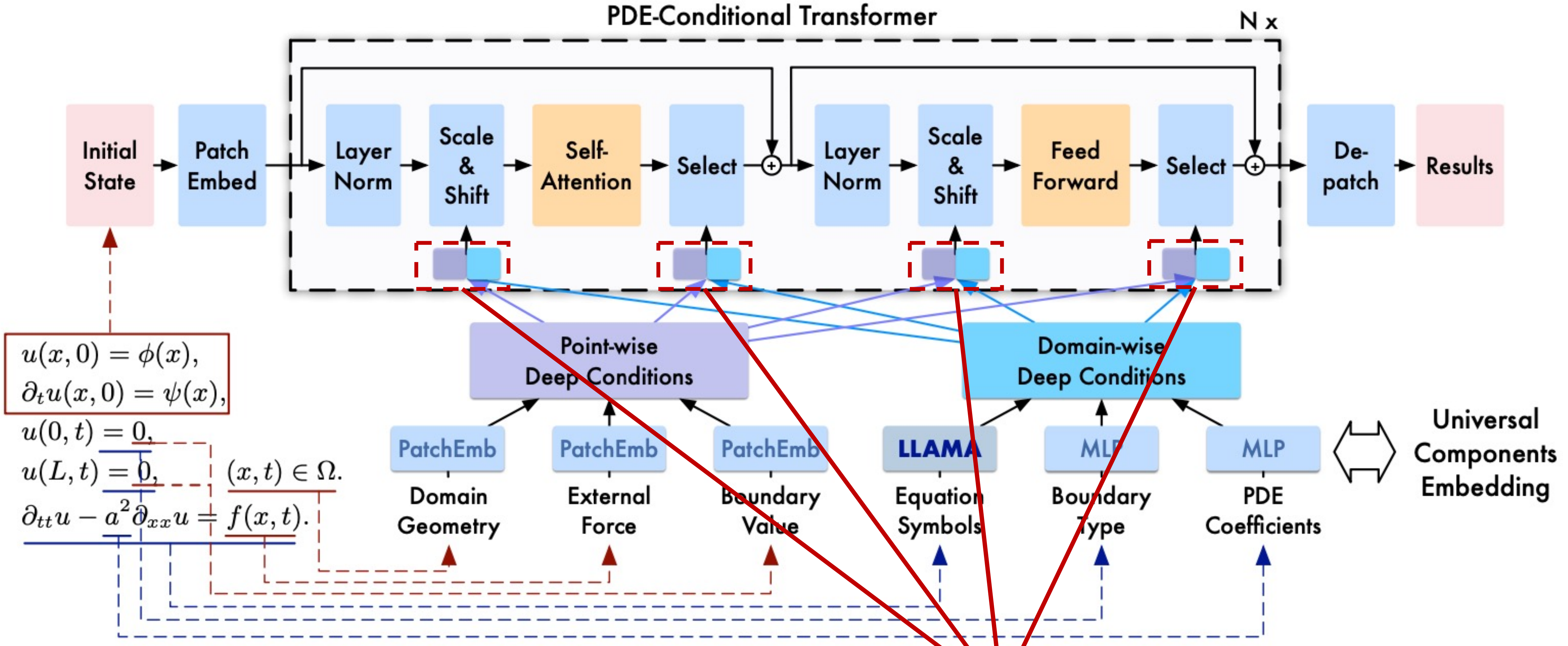
- ☐ Embedding of other components
  - ✓ **Domain-wise** components are embedded by a 2-layer MLP
  - ✓ **Point-wise** components are patchified and embedded into tokens
- ☐ Deep condition consolidation
  - ✓ Embedded conditions of each categories are aggregated together
  - ✓ Make for easy adaptation to novel PDE components in downstream tasks

PDE-Conditional Transformer



① Unify Embedding ② Condition Aggregation

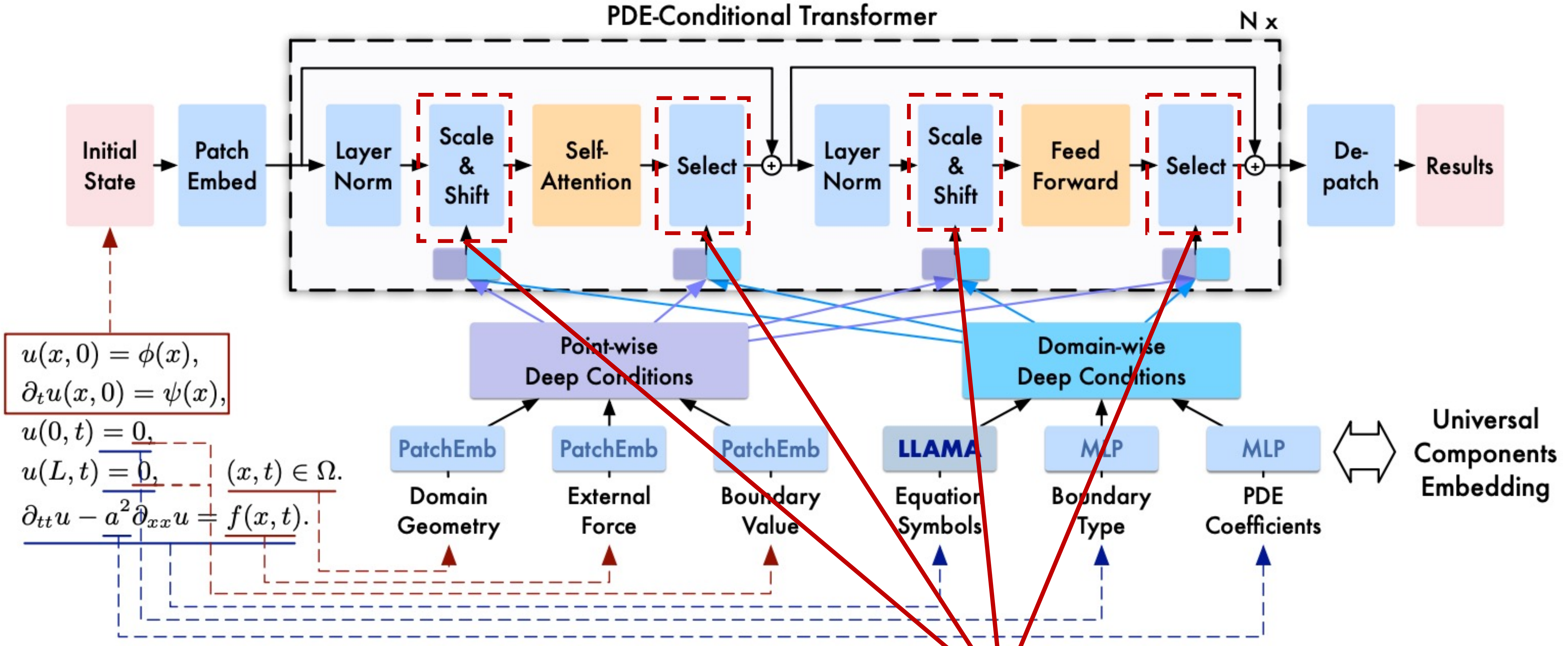
PDE-Conditional Transformer



① Unify Embedding

$$\mathrm{I_{op}} = \mathrm{Concat}(\mathrm{MLP}(\mathrm{C_{domain}}).\mathrm{repeat}, \mathrm{MLP}(\mathrm{C_{point}}))$$
$$\mathrm{I'_{op}} = \mathrm{Concat}(\mathrm{MLP}(\mathrm{C_{domain}}).\mathrm{repeat}, \mathrm{MLP}(\mathrm{C_{point}}))$$

PDE-Conditional Transformer



② Condition Aggregation

$$X^{n-1/2} = I_{select} \odot \mathrm{Attn}(I_{scale} \odot \mathrm{LayerNorm}(X^{n-1}) + I_{shift}) + X^{n-1}$$
$$X^{n} = I'_{select} \odot \mathrm{FFN}\big(I'_{scale} \odot \mathrm{LayerNorm}(X^{n-1/2}) + I'_{shift}\big) + X^{n-1/2}$$

Experiments

Benchmarks	#Dim	#Resolution	# Samples	#Size	Symbols	Coefficient	Force	Geometry	Boundary
HeterNS	2D+Time	(64,64,10)	15k	4.6 GB	×	✓	✓	×	×
PDEformer [45]	1D+Time	(256,100)	3M	300 GB	✓	✓	✓	×	✓
DPOT [10]	2D+Time	(128,128,10)	74.1k	384 GB	✓	✓	✓	✓	✓

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

Heterogeneous 2D Navier-Stokes Equation

- Generalize to unseen coefficients

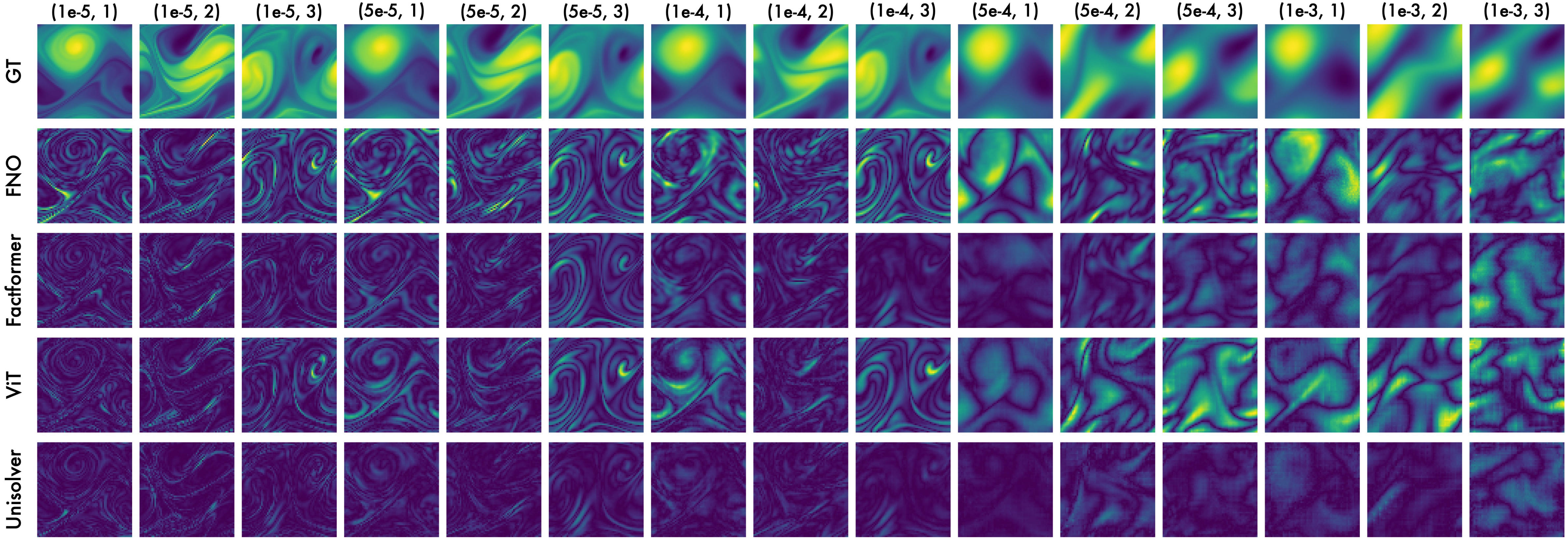
HeterNS	Viscosity ν	8e-6	1e-5	3e-5	5e-5	8e-5	1e-4	3e-4	5e-4	8e-4	1e-3	2e-3
	Params	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD
FNO	4.7M	0.0702	0.0669	0.0373	0.0225	0.0141	0.0114	0.0088	0.0031	**0.0084**	0.0011	0.2057
Factformer	9.4M	0.0489	0.0438	0.0489	0.0128	0.0297	0.0064	0.1386	0.0018	0.0631	0.0010	0.3207
ViT	4.8M	0.0458	0.0432	0.0353	0.0206	0.0119	0.0098	0.0100	0.0031	0.0174	0.0015	0.1878
Unisolver	4.1M	**0.0336**	**0.0321**	**0.0178**	**0.0094**	**0.0064**	**0.0051**	**0.0066**	**0.0015**	0.0096	**0.0008**	**0.1504**
Promotion	-	26.6%	25.7%	49.6%	26.6%	46.2%	20.3%	34.0%	16.7%	-	20.0%	19.9%

- Generalize to unseen external force

HeterNS	Force ω	0.5	1	1.5	2	2.5	3	3.5
	Params	OOD	ID	OOD	ID	OOD	ID	OOD
FNO	4.7M	1.110	0.0640	0.1742	0.0661	0.1449	0.1623	0.2974
Factformer	9.4M	0.9998	0.0326	0.1110	0.0438	0.1243	0.0803	0.2257
ViT	4.8M	0.7900	0.0348	0.1412	0.0432	0.1240	0.1000	0.2080
Unisolver	4.1M	**0.0980**	**0.0244**	**0.0770**	**0.0321**	**0.0720**	**0.0720**	**0.1740**
Promotion	-	87.6%	25.2%	30.6%	25.7%	41.9%	10.3%	16.4%

Heterogeneous 2D Navier-Stokes Equation

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

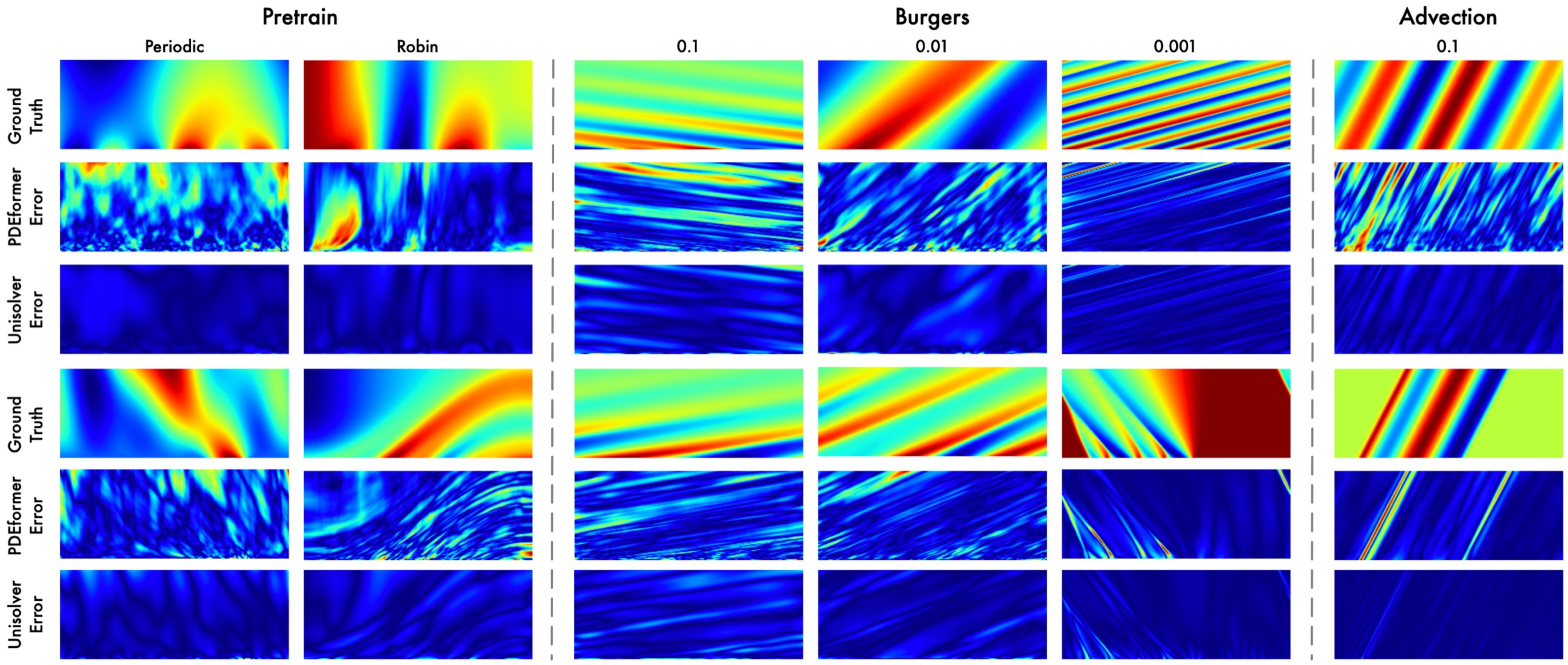


1D Time-dependent PDEs proposed by PDEformer

- Models are pretrained on a dataset with **3 million 1D PDEs** with varied **coefficients, external force, boundary conditions and equation symbols**
- Then tested on **OOD downstream PDE datasets** generated by PDEBench, including the Burgers equation and the advection equation

1D Time-dependent PDEs	Tasks	Pretrain		Burgers			Advection
	Params	Periodic	Robin	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\beta = 0.1$
PDEformer-L	22M	0.0211	0.0238	0.00744	0.0144	0.0393	0.0178
Unisolver	19M	**0.0107**	**0.0108**	**0.00513**	**0.00995**	**0.0299**	**0.0138**
Promotion	-	49.3%	54.6%	31.0%	30.9%	23.9%	22.5%
PDEformer-L (FT-100)	22M	-	-	0.00364	0.0112	0.0331	0.00975
Unisolver (FT-100)	19M	-	-	**0.00105**	**0.00474**	**0.0170**	**0.00420**
Promotion	-	-	-	71.2%	57.7%	48.6%	59.6%

Showcases



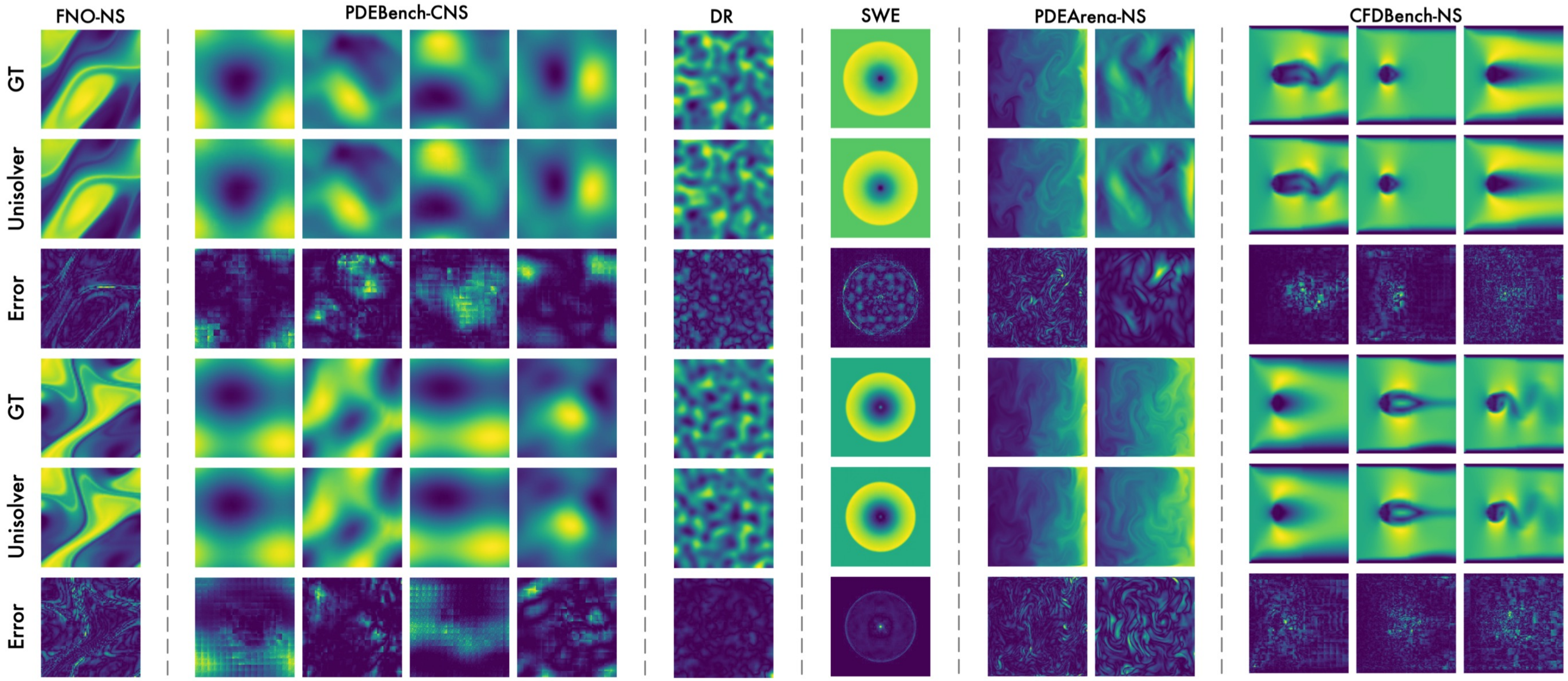
Pretrain
Burgers
Advection
Periodic
Robin
0.1
0.01
0.001
0.1
Ground Truth
PDEformer Error
Unisolver Error
Ground Truth
PDEformer Error
Unisolver Error

2D Mixed PDEs proposed by DPOT

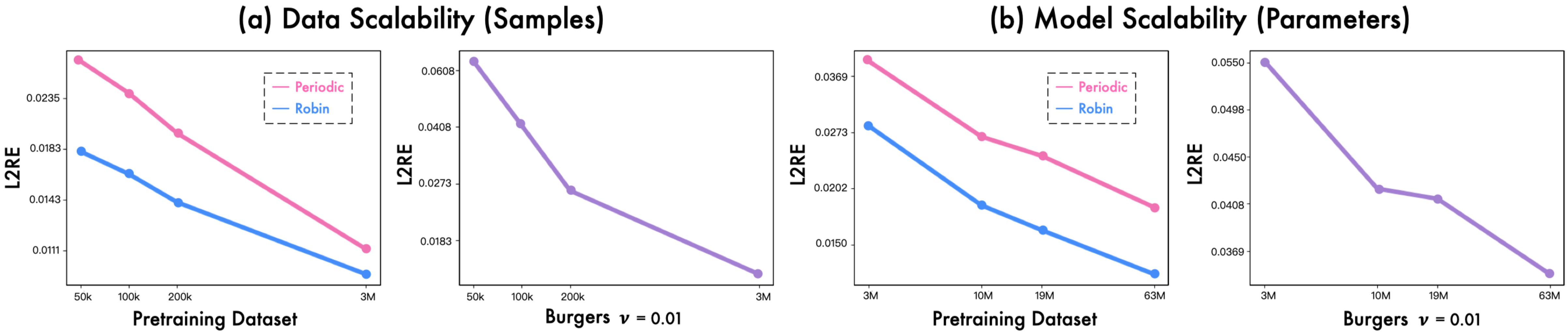
The models are pretrained on 12 datasets collected by DPOT, with varied coefficients, external force, boundary conditions and geometries

2D Mixed PDEs	Equations Params	FNO-NS-ν 1e-5	1e-4	1e-3	PDEBench-CNS-(M,ζ) (1, 0.1)	(1, 0.01)	(0.1, 0.1)	(0.1, 0.01)	DR -	SWE -	PDEArena-NS -	Force	CFDBench-NS Geometry	Mean -
DPOT-S	30M	5.53	4.42	1.31	1.53	3.37	1.19	1.87	**3.79**	0.66	9.91	31.6	0.70	5.50
Unisolver	33M	**4.17**	**3.36**	**0.61**	**1.23**	**2.89**	**1.01**	**1.59**	4.39	**0.45**	**6.87**	**27.4**	**0.54**	**4.54**
Promotion	-	24.6%	24.0%	53.4%	19.6%	14.2%	15.1%	15.0%	-	31.8%	30.7%	13.3%	22.9%	17.5%
DPOT-S-FT	30M	4.49	3.42	0.68	1.52	2.11	1.50	1.51	1.71	0.22	8.92	29.0	**0.44**	4.63
Unisolver-FT	33M	**3.82**	**2.79**	**0.31**	**0.95**	**1.99**	**1.01**	**1.34**	**1.37**	**0.20**	**6.67**	**26.8**	0.47	**3.98**
Promotion	-	14.9%	18.4%	54.4%	37.5%	5.69%	32.6%	11.3%	19.8%	9.1%	25.2%	7.6%	-	14.0%

Showcases


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

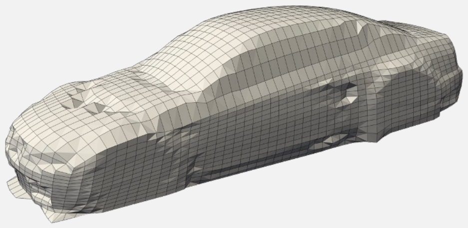
Scalability



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

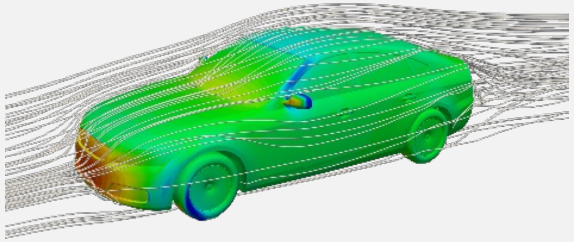
A Roadmap to Practical Neural PDE Solvers

Industrial simulation with CAE



Varied Geometries

Ansys
DASSAULT SYSTEMES
SIEMENS
HEXAGON



Physical Simulation

Neural PDE Solver (Our work)

Q1: High-dimensional Mapping Approximation

Q2: Large-scale Irregular Meshes

Q3: Generalization among varied PDEs

Deep Models → Latent Spectral Models → Transolver → Unisolver → ...

A Roadmap to Practical Neural PDE Solvers

Mingsheng Long

School of Software, Tsinghua University

July 2024