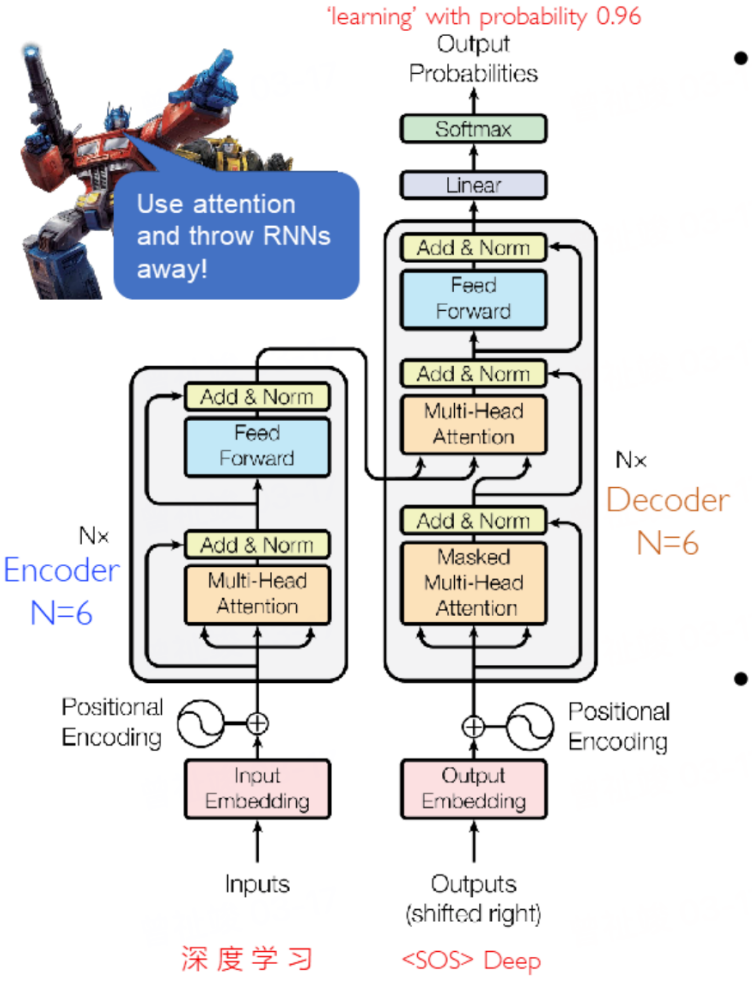
Solving Partial Differential Equation By Transformer

BY ZHIJUN ZENG

2024年3月28日

1 Recap Of Transformer

Transformer



- Main components:
  - ① Scaled Dot-Product Attention
  - ② (Masked) Multi-Head Attention
  - ③ Position-wise FFN
  - ④ Residual Connections
  - ⑤ Layer Normalization
  - ⑥ Positional encoding
- Encoder-decoder architecture:
  - ① Encoder
  - ② Decoder with Masking
  - ③ Encoder-Decoder Attention

① Scaled Dot-Product Attention

- Recall relevance $e_{ij} = v_a^{\mathrm{T}} \tanh\left(W_a s_{i-1} + U_a h_j\right)$: too complex!
- Given query q (to match others) and key k (to be matched):
  - **Bilinear**

$$a(q,k) = q^T W k \longrightarrow \text{Parameter-heavy}$$

  - **Dot Product**

$$a(q,k) = q^T k \longrightarrow \text{Dimension-sensitive}$$

  - **Scaled Dot-Product**

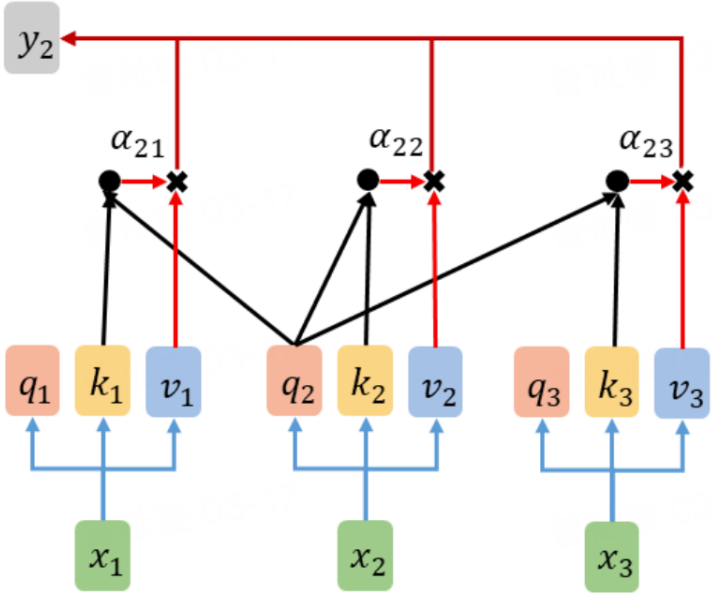
$$a(q,k) = q^T k / \sqrt{d_k} \longrightarrow \text{Dimension of } q \text{ and } k$$

  - For independent q_i, k_i with mean 0 and variance 1:
  - Dot-product $q^T k = \sum_{i=1}^{d_k} q_i k_i$ has mean 0 and variance d_k.
    - »Will not cause large values going into the saturation of Softmax

① Scaled Dot-Product Attention

$$y_1 = \sum_i \alpha_{1i} v_i \qquad \alpha_{1i} = \frac{\exp(q_1 k_i)}{\sum_j \exp(q_1 k_j)}$$

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

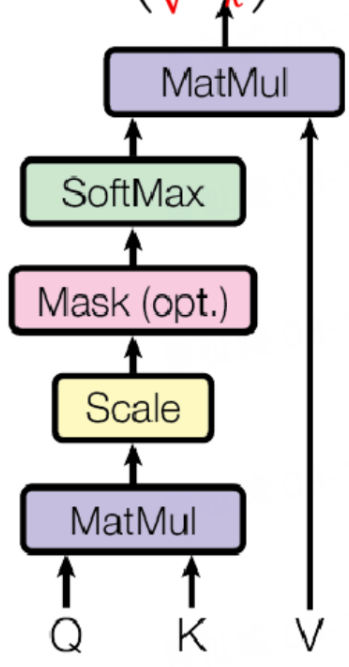


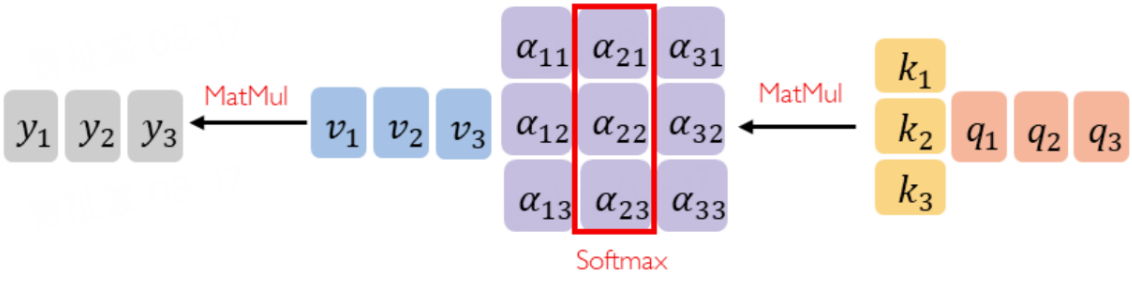
$$\begin{matrix} q_1 & q_2 & q_3 \end{matrix} = W^q \begin{matrix} x_1 & x_2 & x_3 \end{matrix} \quad Q$$

$$\begin{matrix} k_1 & k_2 & k_3 \end{matrix} = W^k \begin{matrix} x_1 & x_2 & x_3 \end{matrix} \quad K$$

$$\begin{matrix} v_1 & v_2 & v_3 \end{matrix} = W^v \begin{matrix} x_1 & x_2 & x_3 \end{matrix} \quad V$$

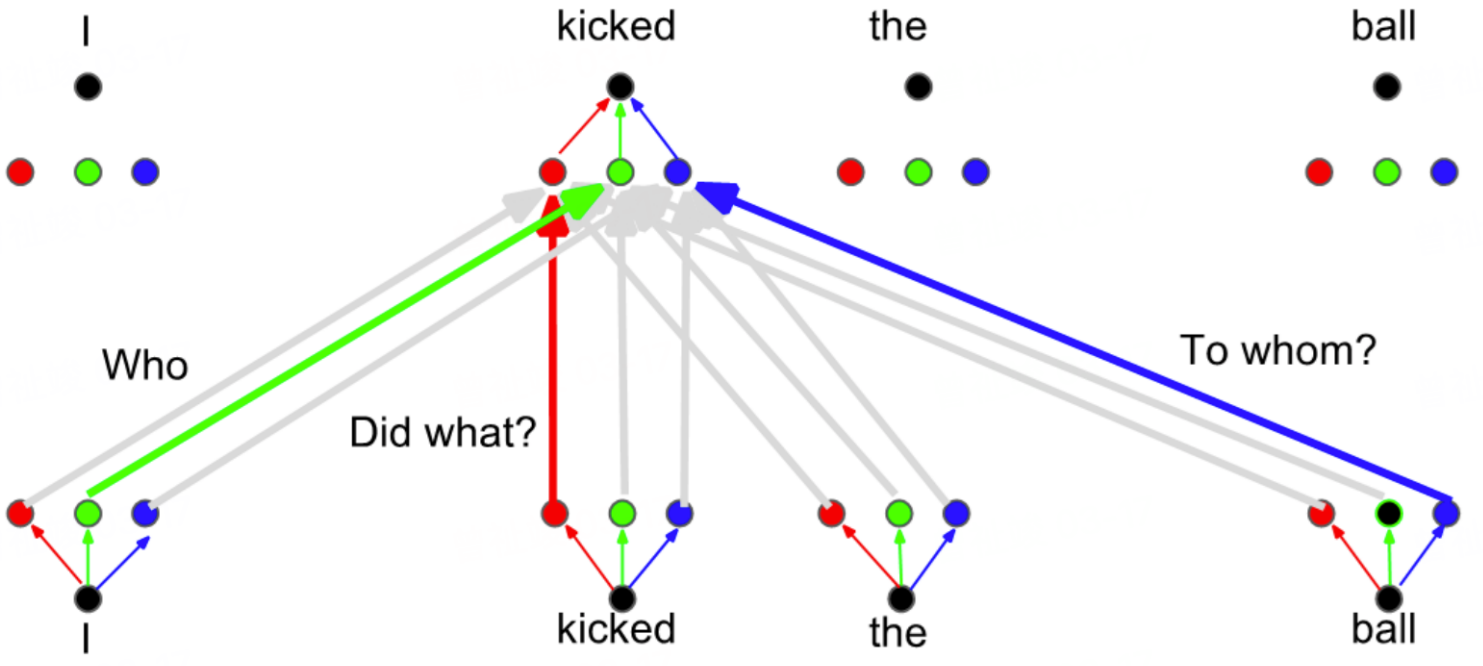
$$q_i = W^q x_i$$
$$k_i = W^k x_i$$
$$v_i = W^v x_i$$





Ignore $\sqrt{d_k}$ for simplicity

② Multi-Head Attention



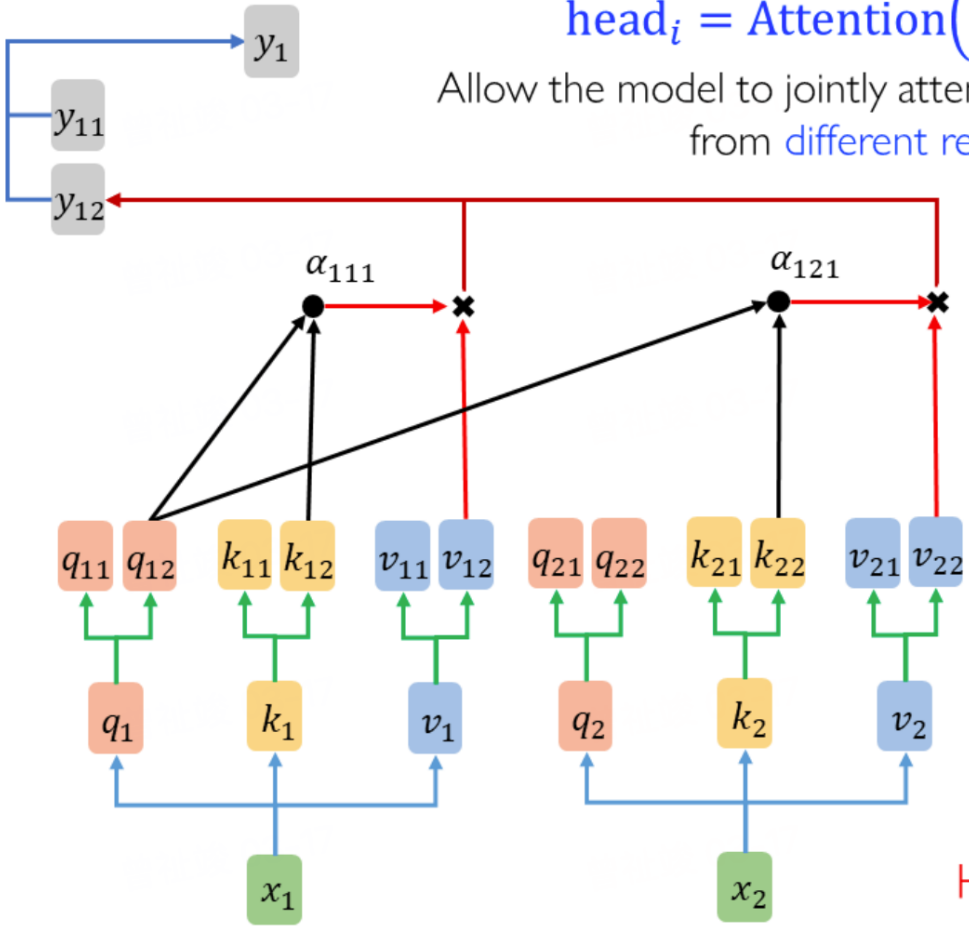
Different heads extract different information (like channels in CNNs)

② Multi-Head Attention

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)W^O$$

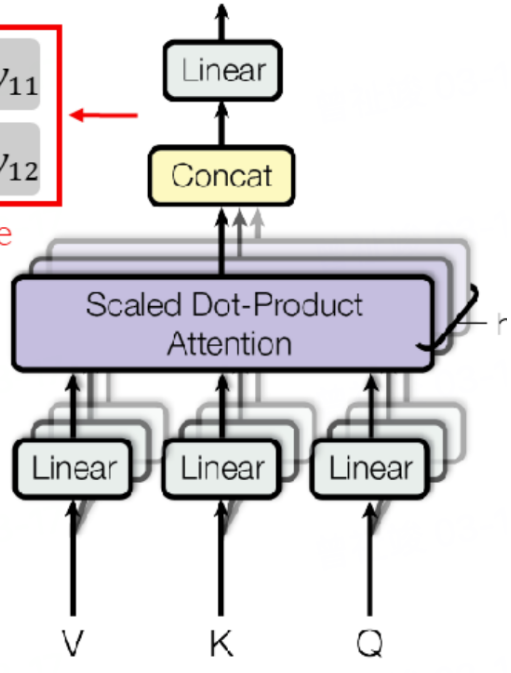
$$\text{head}_i = \text{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right)$$

Allow the model to jointly attend to information at different positions from different representation subspaces.



$y_1 = W^0 \begin{matrix} y_{11} \\ y_{12} \end{matrix}$

Maintain shape

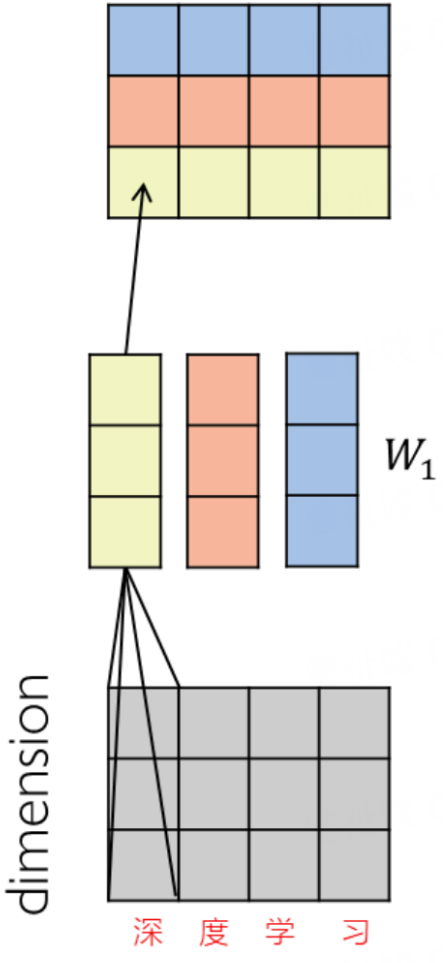


Highly parallelizable, but high memory

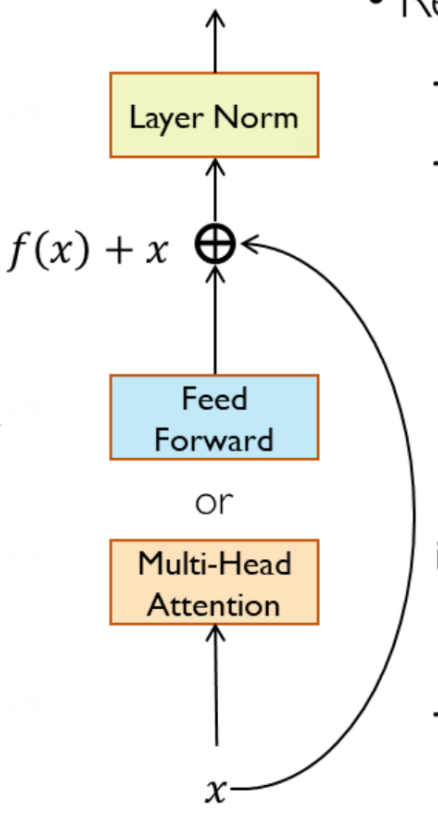
③ Position-wise FFN

- Apply a two-layer position-wise MLP
  - Consisting of two linear transformations with a ReLU activation in between

    $\mathbf{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$
  - Huge number of parameters.
- Equivalent to apply $\mathbf{1D}$ convolution layers
  - Recall 1×1 convolution
  - Parameter are sharing among all positions
  - Applied to sequences with arbitrary length



④ Residual Connection

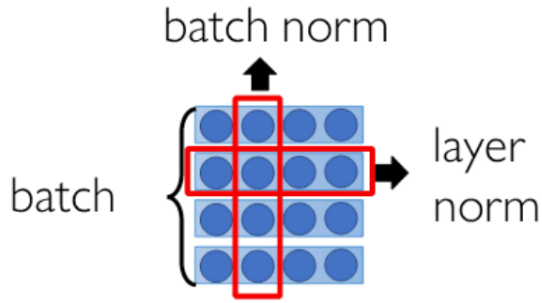


- Residual connection:
  - Stronger gradients
  - Help remember positional encoding

- Residual has been de facto design everywhere!

⑤ Layer Normalization

- Layer normalization:
  - Center embeddings around origin, which helps attention layers
  - No train-inference mismatch: Same computation throughout

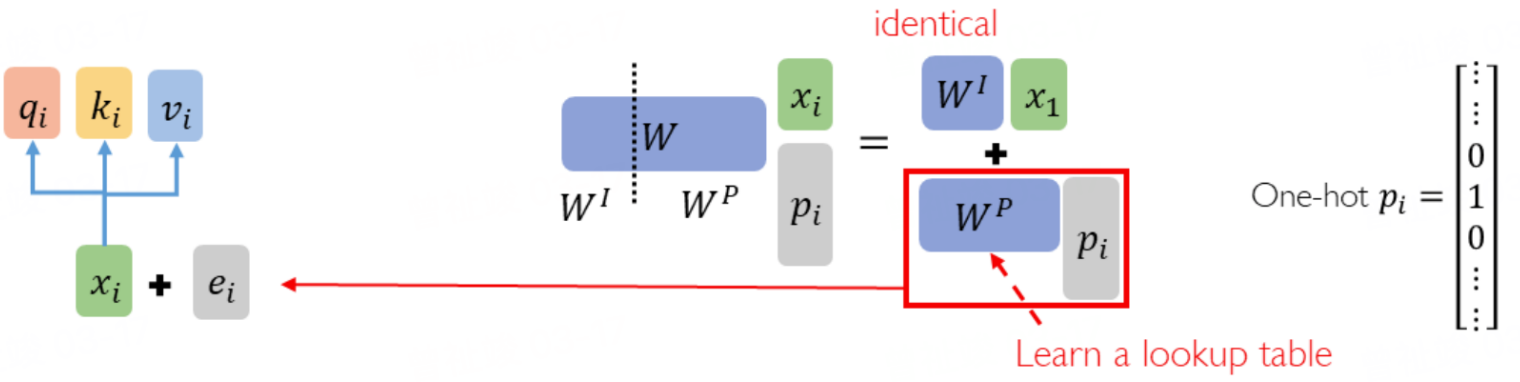


  - DeepNet: Scaling Transformers to 1000 Layers. arXiv 2022. 😄

⑥ Positional Encoding

- No position information in self-attention 😔

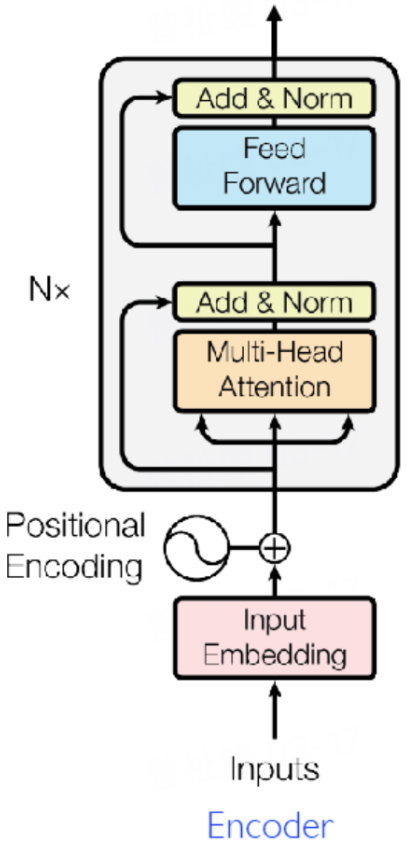
Inject some information about the relative or absolute position of the tokens in the sequence



- Each position i has a unique positional vector e_i yielded by fixed function

$$e_i(2j) = \sin\left(\frac{i}{10000^{\frac{2j}{d_{\text{model}}}}}\right), e_i(2j+1) = \cos\left(\frac{i}{10000^{\frac{2j}{d_{\text{model}}}}}\right)$$

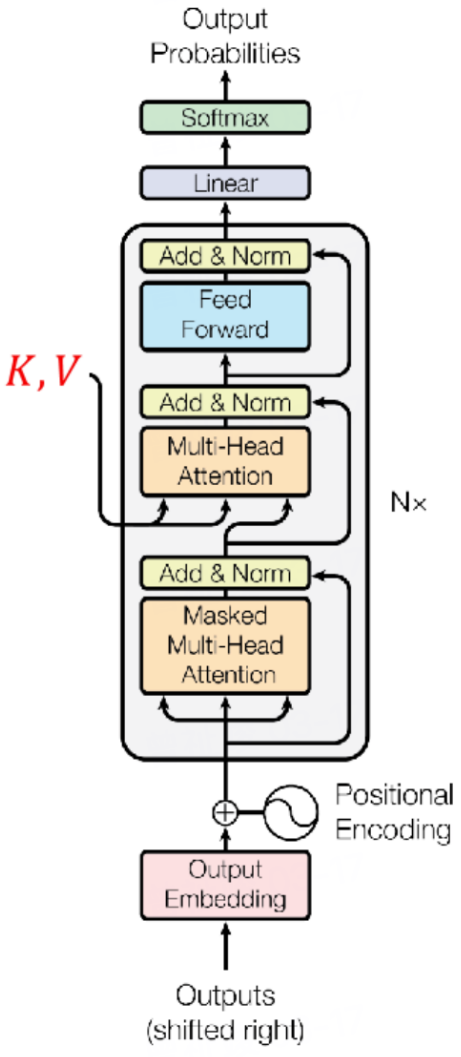
① Encoder



- The encoder stacks N encoder blocks
  - ($N = 6$ in original paper)
  - Multi-head self-attention
  - Position-wise FFN
  - Positional encoding at the bottoms of encoder
- Each block maintains shape
  - (#vectors, vector length)

- Weaknesses:
  - Complexity $O(n^2)$, n is sequence length
  - Difficult to train with huge parameters
  - Many heads are unimportant and can be pruned with little impact on performance

② Decoder



- The decoder stacks N decoder blocks
  - ($N = 6$ in original paper)
  - Multi-head self-attention
    - » K and V in the multi-head self-attention are from the encoder outputs
  - Masked multi-head self-attention
  - Position-wise FFN
  - Positional encoding at the bottoms of decoder
- Each block maintains shape
  - (#vectors, vector length)
- How does decoder work at inference and training?

2 Fourier/Galerkin Transformer

2.1 Attention Mechanism

Then encoder contains a stack of identical simple attention-based layer. For simplicity, we consider a single attention head that maps $\boldsymbol{y} \in \mathbb{R}^{n \times d}$ to another element in $\mathbb{R}^{n \times d}$ and defines trainable projection matrices/latent representation Q,K,V

$$W^Q, W^K, W^V \in \mathbb{R}^{d \times d}, Q := \boldsymbol{y} W^Q, K := \boldsymbol{y} W^K, V := \boldsymbol{y} W^V$$

Shortly, Galerkin/Fourier Attention uses a mesh-weighted normalization without softmax and allows a scaling to propagate through the encoder

$$\mathrm{Attn}_{\mathrm{sp}} \colon \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times d}, \tilde{\boldsymbol{y}} \leftarrow \boldsymbol{y} + \mathrm{Attn}_{+}(\boldsymbol{y}), \boldsymbol{y} \to \tilde{\boldsymbol{y}} + g(\tilde{\boldsymbol{y}})$$

$$(Fourier) \quad \mathbf{z} = \mathrm{Attn}_{\mathfrak{f}}(\mathbf{y}) := (\tilde{Q}\, \tilde{K}^{\top})\, V / n$$

$$(Galerkin) \quad \mathbf{z} = \mathrm{Attn}_{\mathfrak{g}}(\mathbf{y}) := Q\, (\tilde{K}^{\top} \tilde{V}) / n$$

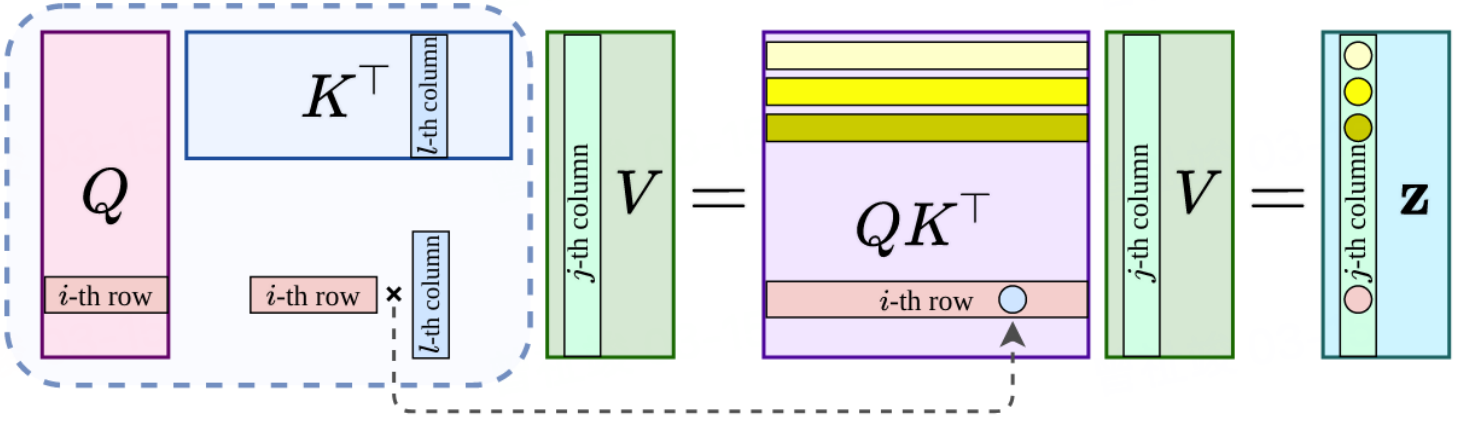
here $\tilde{}$ denotes layernorm and g is standard 2-layer FFN

$$\mathbf{y} \mapsto \mathrm{Ln}(\mathbf{y} + \mathrm{Attn}_{\dagger}(\mathbf{y}) + g(\mathrm{Ln}(\mathbf{y} + \mathrm{Attn}_{\dagger}(\mathbf{y})))), \text{where } \mathrm{Attn}_{\dagger}(\mathbf{y}) := Q\, (K^{\top} V) / n$$

Consider an operator learning problem with domain $\Omega \subset \mathbb{R}$,$\{x_i\}^n$ denotes the set of grid points in the discretized Ω such that the weight $h = \frac{1}{n}$ is the mesh size.

Let $\zeta_q(\cdot), \phi_k(\cdot), \psi_v(\cdot): \Omega \to \mathbb{R}^{1\times d}$ denotes the feature map of Q,K,V, i.e the i-th row of Q,K,V written as $\boldsymbol{q}_i = \zeta_q(x_i), \boldsymbol{k}_i = \phi_k(x_i), \boldsymbol{v}_i = \psi_v(x_i)$.

Using $V \in \mathbb{R}^{n\times d}$ with full column rank as example. its column contains a set of bases $\{v_j(\cdot)\}_{j=1}^d$ evaluated at grid point, similarly to $\{q_j(\cdot)\}_{j=1}^d, \{k_j(\cdot)\}_{j=1}^d, \{z_j(\cdot)\}_{j=1}^d$, and we denotes $\boldsymbol{v}^j$ as the j-th column of V ,then $(\boldsymbol{v}^j)_l = v_j(x_l)$

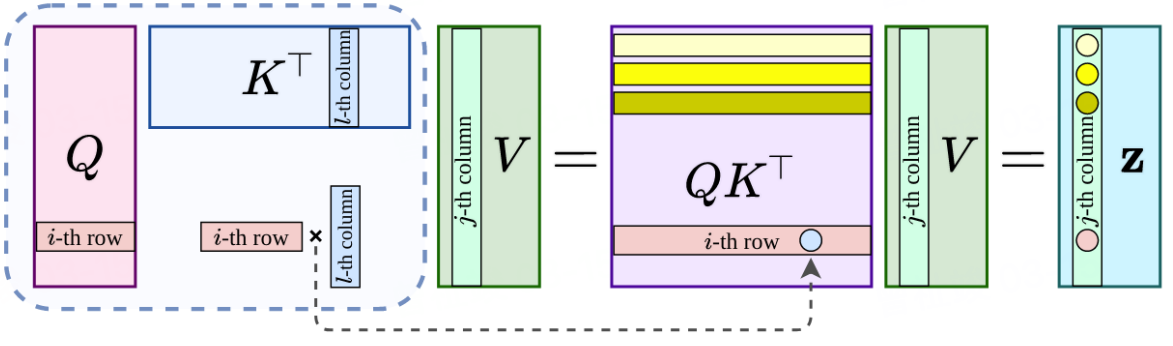


2.2 Fourier-type attention with quadratic complexity

In the Fourier-type attention, Q, K are assumed to be normalized for simplicity, the j-th column in the i-th row of z is computed by

$$
\begin{aligned}
(\boldsymbol{z}_i)_j =& h\left(Q\,K^\top\right)_i \cdot \boldsymbol{v}^j = h\left(\boldsymbol{q}_i \cdot \boldsymbol{k}_1, \ldots, \boldsymbol{q}_i \cdot \boldsymbol{k}_l, \ldots, \boldsymbol{q}_i \cdot \boldsymbol{k}_n\right)^\top \cdot \boldsymbol{v}^j \\
=& h \sum_{l=1}^{n} \left(\boldsymbol{q}_i \cdot \boldsymbol{k}_l\right)\left(\boldsymbol{v}^j\right)_l \approx \int_\Omega \left(\zeta_q(x_i) \cdot \phi_k(\xi)\right) v_j(\xi) \mathrm{d}\xi
\end{aligned}
$$

where the h- weight facilitates the numerical quadrature interpretation of the inner product. Concatenating columns $1 \leqslant j \leqslant d$ yeilds the $i-\text{row}$ $\boldsymbol{z}_i$ of output $\boldsymbol{z}$: $\boldsymbol{z}_i \approx \int_\Omega \left(\zeta_q(x_i) \cdot \phi_k(\xi)\right) \psi_v(\xi) \mathrm{d}\xi$, therefore the output at i-th row computes approximately an integral transform of a non-symmetric learnable kernel $\kappa(x, \xi) = \zeta_q(x)\,\phi_k(\xi)$ evaluated at x_i.



2.3 Galerkin-type attention with linear complexity

For Galerkin-type attention, K,V are assumed to be normalized, we consider the i-th entry in j-th column of z

$$(\boldsymbol{z}^j)_i = h\boldsymbol{q}_i^\top \cdot (K^\top V)_{\cdot j},$$

which is the inner product of the i-th row of Q and the j-th column of $K^T V$. Thus,

$$\boldsymbol{z}^j = h\begin{pmatrix} | & | & | & | \\ \boldsymbol{q}_1 & \boldsymbol{q}_2 & \cdots & \boldsymbol{q}_n \\ | & | & | & | \end{pmatrix}^\top (K^\top V)\cdot j = h\left((K^\top V)_{\cdot j}^\top \begin{pmatrix} - & \boldsymbol{q}^1 & - \\ - & \vdots & - \end{pmatrix}\right)^\top$$

$(K^\top V)\cdot j$ ontains the coefficients for the linear combination of the vector representations$\{q^l\}_{l=1}^d$ of the bases stored in Q's column space to form the output z

$$\boldsymbol{z}^j = h\sum_{l=1}^{d} \boldsymbol{q}^l\,(K^\top V)_{lj}, \quad \text{where } (K^\top V)_{\cdot j} = (\boldsymbol{k}^1\cdot\boldsymbol{v}^j, \boldsymbol{k}^2\cdot\boldsymbol{v}^j, \cdots, \boldsymbol{k}^d\cdot\boldsymbol{v}^j)^\top$$

$$z_j(x_i) := (\boldsymbol{z}^j)_i = h\sum_{l=1}^{d} (\boldsymbol{k}^l\cdot\boldsymbol{v}^j)(\boldsymbol{q}^l)_i \approx \sum_{l=1}^{d}\left(\int_\Omega v_j(\xi)\,k_l(\xi)\mathrm{d}\xi\right) q_l(x_i)$$

	Example 1: $n = 8192$				Encoders only: $n = 8192, d = 128, l = 10$				Computational complexity of the dot-product per layer
	Mem	CUDA Mem	Speed	GFLOP	Mem	CUDA Mem	Speed	GFLOP	
ST	18.39	31.06	5.02	1393	18.53	31.34	4.12	1876	$O(n^2 c_e d)$
FT	10.05	22.92	6.10	1138	10.80	22.32	5.46	1610	$O(n^2 d)$
LT	2.55	2.31	12.70	606	2.73	2.66	10.98	773	$O(n(d^2 + c_e d))$
GT	**2.36**	**1.93**	**27.15**	**275**	**2.53**	**2.33**	**19.20**	**412**	$O(nd^2)$

Table 2: (a) Evaluation relative error ($\times 10^{-3}$) of Burgers' equation 5.1. (b) Evaluation relative error ($\times 10^{-2}$) of Darcy interface problem 5.2.

(a)

	$n = 512$	$n = 2048$	$n = 8192$
FNO1d [57]	15.8	14.6	13.9
FNO1d `1cycle`	4.373	4.126	4.151
FT regular Ln	1.400	1.477	1.172
GT regular Ln	2.181	1.512	2.747
ST regular Ln	1.927	2.307	1.981
LT regular Ln	1.813	1.770	1.617
FT Ln on Q, K	**1.135**	**1.123**	**1.071**
GT Ln on K, V	**1.203**	**1.150**	**1.025**
ST Ln on Q, K	1.271	1.266	1.330
LT Ln on K, V	1.139	1.149	1.221

(b)

	$n_f, n_c = 141, 43$	$n_f, n_c = 211, 61$
FNO2d [57]	1.09	1.09
FNO2d `1cycle`	1.419	1.424
FT regular Ln	**0.838**	**0.847**
GT regular Ln	0.894	0.856
ST regular Ln	1.075	1.131
LT regular Ln	1.024	1.130
FT Ln on Q, K	0.873	0.921
GT Ln on K, V	**0.839**	**0.844**
ST Ln on Q, K	0.946	0.959
LT Ln on K, V	0.875	0.970

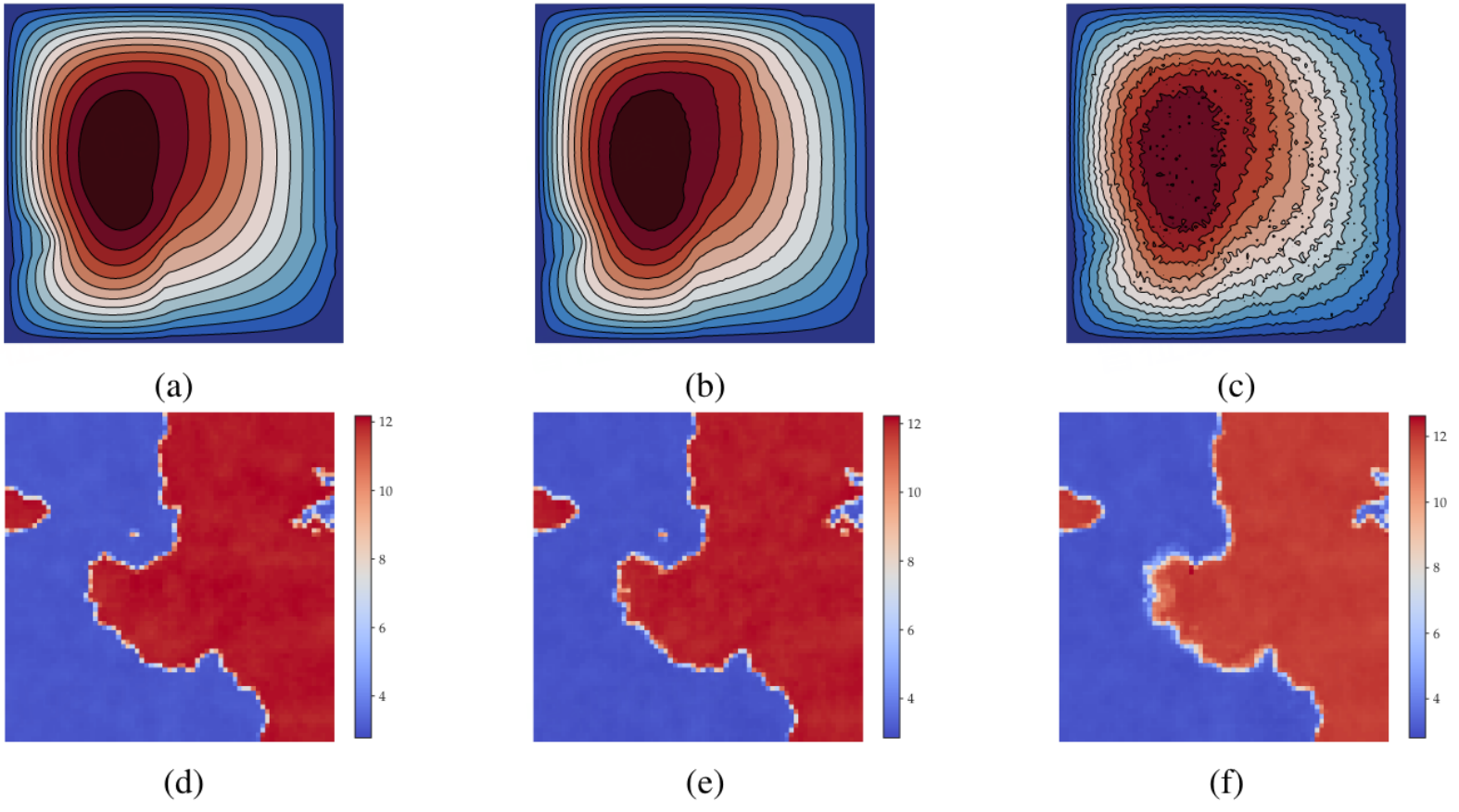


Figure 11: Galerkin transformer evaluation for the inverse interface coefficient identification problem using the same sample with Figure 10, model trained and evaluated under the same amount of noises: (a)–(c) the input $u_h(x)$ with noise level 0, 1%, and 10% on 211×211 grid; (d)–(f) the recovered coefficient through evaluation with noise level 0, 1%, and 10% on 71×71 grid with relative error being 0.0160, 0.0292, and 0.0885, respectively.

3 Operator Transformer

3.1 Cross Attention

Besides the standard attention (Galerkin Type or Fourier Type), the Q, K, V are computed on the same grid of input function. (Consider sound speed to wavefield). When decoding, one need to evaluate the solution on given point $\{y_i\}_{i=1}^m$, one can use cross attention. Cross attention can be viewed as $\{k_l(\cdot), v_l(\cdot), q_l(\cdot)\}_{l=1}^d$, where k, v are defined on the input grid $D(x)$:$\{x_i\}_{i=1}^n$, q is defined on the output grid $D(y)$:$\{y\}_{i=1}^m$

$$z_s(y_j) = \sum_{l=1}^{d} \frac{\sum_{l=1}^{n} k_l(x_i)\, v_s(x_i)}{n} q_l(y_j)$$

we add FFN to improve the nonlinearity of NN.

3.2 Encoder

Input encoder inputs the sampled value $a(x_i)$ of $\{x_i\}_{i=1}^n$. After point-wise MLP, these values are map to latent space $f^{(0)}$, then input encoder use self-attention block to extract feature of input

$$\mathbf{f}^{(l')} = \mathrm{LayerNorm}(\mathbf{f}^{(l)} + \mathrm{Attn}(\mathbf{f}^{(l)})), \quad \mathbf{f}^{(l+1)} = \mathrm{LayerNorm}(\mathbf{f}^{(l')} + \mathrm{FFN}(\mathbf{f}^{(l')}))$$

Query encoder using a point-wise MLP with Random Fourier Projection to extract feature of query point Y

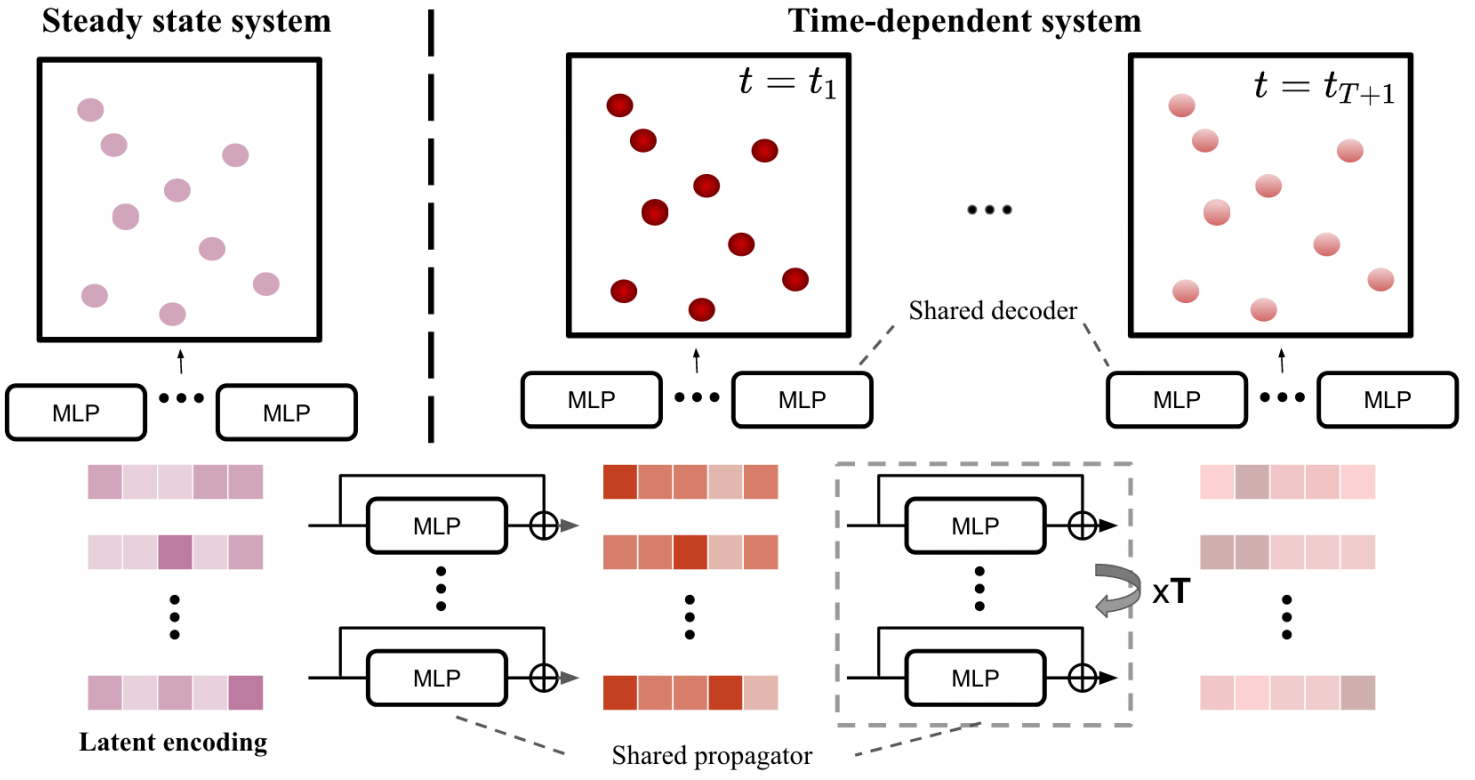
$$\gamma(Y) = [cos\,(2\pi YB), sin\,(2\pi YB)]$$

where $Y = [y_1, \ldots, y_n]^T$ are coordinates and $B \in \mathbb{R}^{d_1 \times d_2}$ sampled from $\mathcal{N}(0, \sigma^2)$

Cross Attention module combines the encoding output $f^{(L)}$ with the query encoder output z^0 to obtain the final output

$$\mathbf{z}' = \mathbf{z}^{(0)} + \mathrm{Cross-Attn}(\mathbf{z}^{(0)}, \mathbf{f}^{(L)}), \quad \mathbf{z} = \mathbf{z}' + \mathrm{FFN}(\mathbf{z}')$$

Also to encode the spatial information Operator Transformer use RoPE.


Steady state system
Time-dependent system
$t = t_1$
$t = t_{T+1}$
Shared decoder
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
MLP
xT
Latent encoding
Shared propagator

Data settings		Relative L_2 norm				
Case	ν, T	FNO-2D	FNO-3D	MWT	G.T.*	OFormer
NS1	$1 \times 10^{-3}, 50$	0.0128	0.0086	**0.0062**	0.0098	0.0104 $\pm$ 0.0005
NS2-part	$1 \times 10^{-4}, 30$	0.1559	0.1918	0.1518	**0.1399**	0.1755 $\pm$ 0.0059
NS2-full	$1 \times 10^{-4}, 30$	0.0834	0.0820	0.0667	0.0792	**0.0645** $\pm$ **0.0011**
NS3	$1 \times 10^{-5}, 20$	0.1556	0.1893	0.1541	**0.1340**	0.1705 $\pm$ 0.0007
NS-mix*	$[0.4, 1] \times 10^{-4}$, 30	0.1650	0.1654	**0.1267**	0.1462	0.1402 $\pm$ 0.0016
# of parameters (M)		0.41/2.37	6.56	179.18	1.56	1.85

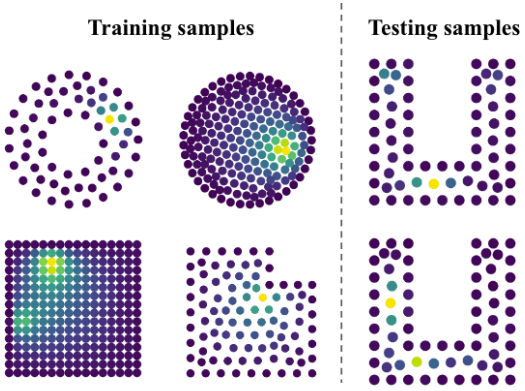
Table 1: Benchmark on 2D Navier-Stokes equation with fixed 64×64 grid. For large dataset (NS-mix) we use a larger version of FNO-2D by increasing modes and width. We adopt Galerkin Transformer (G.T.) with a larger hidden dimension ($64 \rightarrow 96$) to match the size of other models.

Res.	Relative L_2 norm ($\times 10^{-3}$)				
	FNO*	FNO+*	MWT	F.T.	OFormer
512	3.15	**0.72**	1.85	1.14	1.42 $\pm$ 0.03
2048	3.07	**0.72**	1.86	1.12	1.30 $\pm$ 0.07
8192	2.76	**0.70**	1.78	1.07	1.54 $\pm$ 0.10

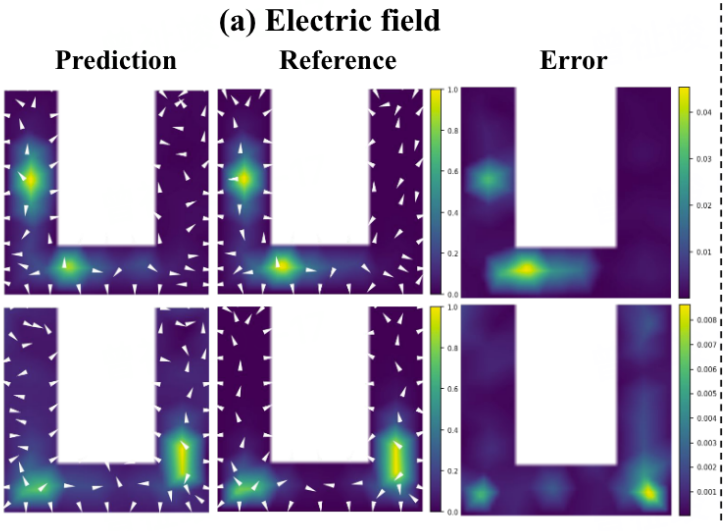
Table 2: Benchmark on 1D Burgers' equation with different resolution. FNO uses ReLU (Nair & Hinton, 2010) whereas FNO+ uses GELU. Both FNO variants remove normalization compared to the original paper.

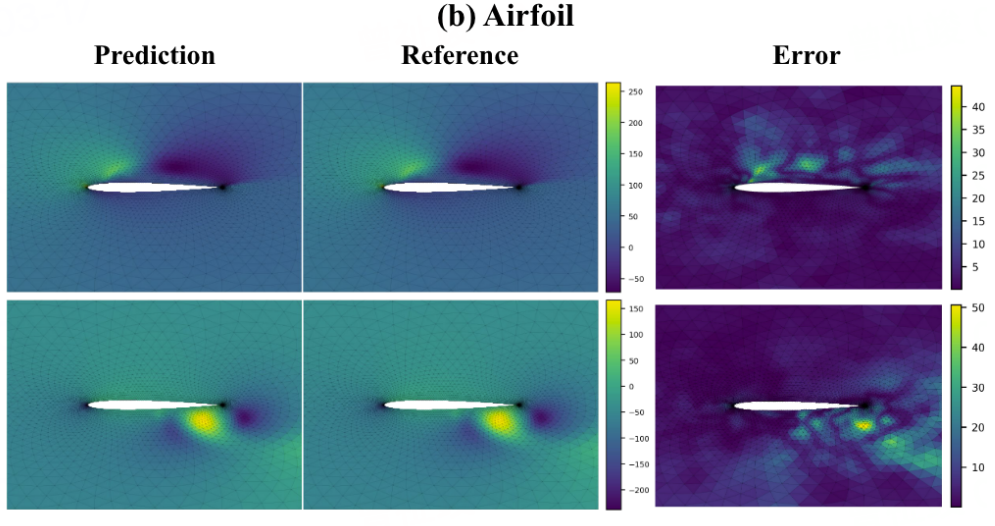
Res.	Relative L_2 norm ($\times 10^{-2}$)		
	FNO	G.T.	OFormer
141	1.09	**0.84**	1.26 $\pm$ 0.03
211	1.09	**0.84**	1.28 $\pm$ 0.02

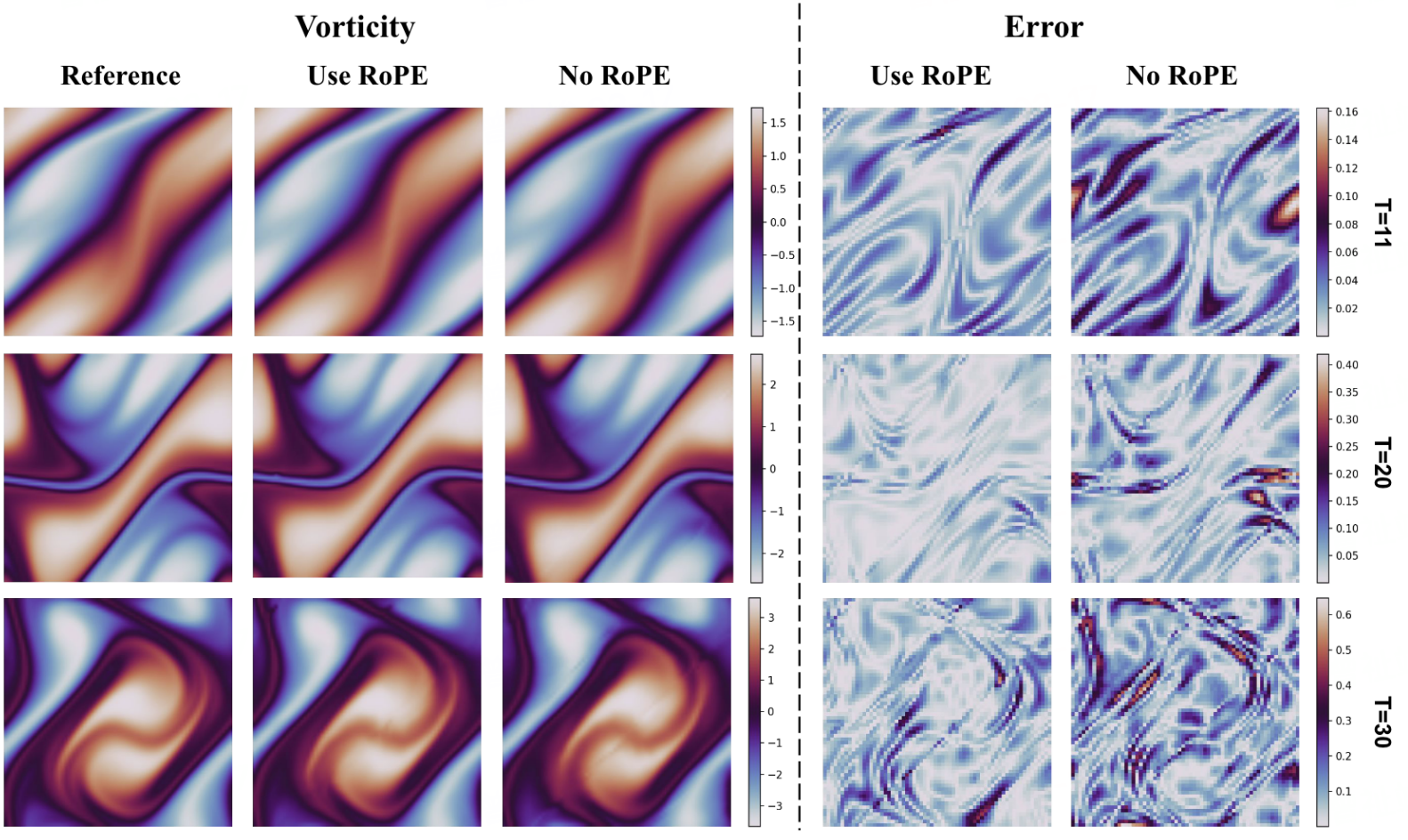
Table 3: Benchmark on 2D Darcy flow with different resolution. MWT's result is not reported since its implementation requires resolution to be power of 2 and thus it is trained on a different data.



Mesh aug.	Model	Mean squared error ($\times 10^{-3}$)			
		Electric		Magnetic	
		Potential	Field	Potential	Field
No	OFormer	**0.174** ± 0.083	**1.690** ± 0.201	**0.138** ± 0.021	**1.470** ± 0.172
	GNN	0.327	2.723	0.161	2.640
Yes	OFormer	0.019 ± 0.004	**0.328** ± 0.025	0.016 ± 0.002	**0.195** ± 0.009
	GNN	**0.006**	1.821	**0.006**	1.338






Vorticity
Error
Reference
Use RoPE
No RoPE
Use RoPE
No RoPE
T=11
T=20
T=30

4 HIERARCHICAL TRANSFORMER

4.1 Multi-scale PDE

Multiscale PDEs, in a narrower sense, refer to PDEs with rapidly varying coefficients

Multiscale elliptic PDE:

$$\begin{aligned} -\nabla \cdot (a(x) \nabla u(x)) &= f(x) \quad x \in D \\ u(x) &= 0 \qquad x \in \partial D \end{aligned}$$

where $0 < a_{\min} \leqslant a(x) \leqslant a_{\max}$, the solution map is $\mathcal{S}: L^{\infty}(D; \mathbb{R}_+) \to H_0^1(D; \mathbb{R}_+)$ such that $u = \mathcal{S}(a)$. The setup $a(x) \in L^{\infty}$ allows rough coefficients with fast oscillation $a(x) = a(x/\varepsilon)$ and high contrast ratio $\frac{a_{\max}}{a_{\min}} \gg 1$.

Navier-Stokes Equation becomes turbulent due to the simultaneous interaction of a wide range of temporal and spatial scales of motion.

Numerical solution of Helmholtz equation exhibits severe difficulties in the high wave number regime due to the interaction of high-frequency waves and numerical mesh.

4.2 Hierachichal Attention Model

4.2.1 Hierarchical Discretization

Let $\mathcal{I}^{(r)}$ be the finest level index set, such that each index $i=(i_1,\ldots,i_r)\in\mathcal{I}^{(r)}$ denotes the finest level spatial objects such as image pixels.

For any $i=(i_1,\ldots,i_r)\in\mathcal{I}^{(r)}$ and $1\le m\le r$, $i^{(m)}=(i_1,\ldots,i_m)$ represents i's m-th level parent node which is the aggregate of finer level objects. $\mathcal{I}^{(m)}:=\{i^{(m)}: i\in\mathcal{I}^{(r)}\}$ is the m-th level index set.

Each index $i\in\mathcal{I}^{(m)}$ corresponds to a token, which can be characterized e.g. by its position $\mathbf{x}_i^{(m)}$ and a feature vector $\mathbf{f}_i^{(m)}$ with number of channels $C^{(m)}$.

4.2.2 Reduce Operation

The reduce operation defines the map from finer-level features to coarser-level features. The reduce map can be abstractly defined as $\mathbf{f}_i^{(m),t}=\mathcal{R}^{(m)}(\{\mathbf{f}_j^{(m+1),t}\}_{j\in i^{(m,m+1)}})$, which maps the $(m+1)$-th level features with indices in $i^{(m,m+1)}$ to the m-th level feature with index $i^{(m)}$. For simplicity, we use a simple linear layer for the operator $\mathcal{R}^{(m)}$ in our current implementation, namely, $\mathbf{f}_i^{(m),t}=R_0^{(m)}\mathbf{f}_{(i,0)}^{(m+1),t}+R_1^{(m)}\mathbf{f}_{(i,1)}^{(m+1),t}+R_2^{(m)}\mathbf{f}_{(i,2)}^{(m+1),t}+R_3^{(m)}\mathbf{f}_{(i,3)}^{(m+1),t}$, where $R_0^{(m)},R_1^{(m)},R_2^{(m)},R_3^{(m)}\in\mathbb{R}^{C^{(m-1)}\times C^{(m)}}$ are parametrized by linear layers.

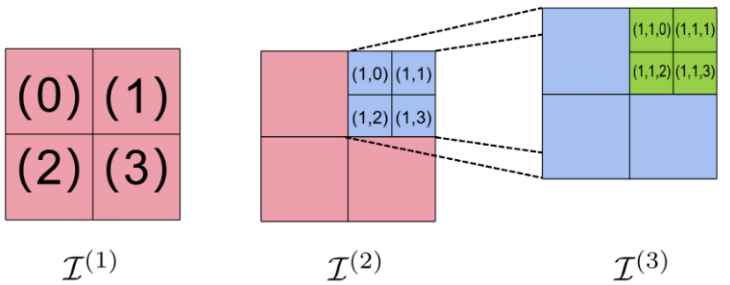


Figure 3.1: Hierarchical discretization and index tree. In this example, the 2D unit square is hierarchically discretized into three levels which are indexed by $\mathcal{I}^{(1)}, \mathcal{I}^{(2)}$ and $\mathcal{I}^{(3)}$, respectively. For example, we denote by $(1)^{(1,2)} = \{(1,0),(1,1),(1,2),(1,3)\}$ the set of the second level child nodes of the node (1).

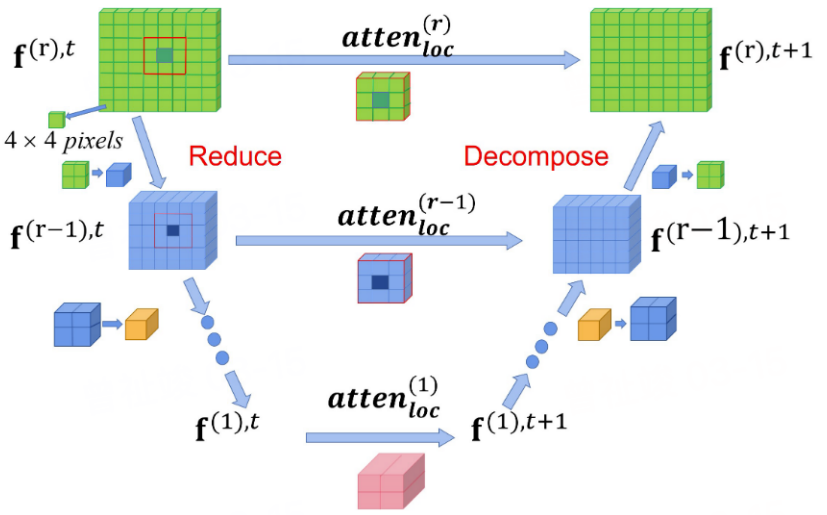


Figure 3.2: One V-cycle of the feature update.

4.2.3 Multilevel Local Aggregation

The finest level features $f_i^{(r),t} \in \mathbb{R}^{C^{(r)}}$ at the evolution step t, with index $i \in \mathcal{I}^{(r)}$, can be updated by the following token aggregation formula through vanilla attention,

$$\text{atten: } f_i^{(r),t+1} = \sum_{j=1}^{N^{(r)}} \mathcal{G}\big(f_i^{(r),t}, f_j^{(r),t}\big)\, v\big(f_j^{(r),t}\big), \quad \text{for } i \in \mathcal{T}^{(r)}, \tag{1}$$

with $N^{(r)} := |\mathcal{I}^{(r)}|$. For simplicity, we ignore the softmax normalizing factor and only consider single-head attention, and we assume that the interaction kernel $\mathcal{G}$ is of the form $\mathcal{G}(f_i, f_j) = \exp\left(W^Q f_i \cdot W^K f_j\right) = \exp\left(q_i \cdot k_j\right)$, where $q_i := W^Q f_i, k_i := W^K f_j$, $v_i := v(f_i) = W^V f_i$, and $W^Q, W^K, W^V \in \mathbb{R}^{C^{(r)} \times C^{(r)}}$ are learnable matrices.

Instead of original attention, we proposed a local aggregation scheme. The local aggregation at the r-th level writes,

$$\text{atten}_{\text{loc}}^{(r)}\text{: } f_i^{(r),t+1} = \sum_{j \in \mathcal{N}^{(r)}(i)} \exp\left(q_i^{(r),t} \cdot k_j^{(r),t}\right) v_j^{(r),t}, \quad \text{for } i \in \mathcal{I}^{(r)}, \tag{2}$$

where $\mathcal{N}^{(r)}(i)$ is the set of the r-th level neighbors of $i \in \mathcal{I}^{(r)}$, $q_i^{(r),t} := \tilde{W}^Q f_i^{(r),t}$, $k_i^{(r),t} := \tilde{W}^K f_i^{(r),t}$, $v_i^{(r),t} := \tilde{W}^V f_i^{(r),t}$ with learnable matrices $\tilde{W}^Q, \tilde{W}^K, \tilde{W}^V \in \mathbb{R}^{C^{(r)} \times C^{(r)}}$. Then for $m = r-1, \ldots, 1$, we calculate the local attention in each level with nested $q_i^{(m),t}$, $k_i^{(m),t}$, $v_i^{(m),t}$

For the m-th level:

$$\text{atten}_{\text{loc}}^{(m)}: f_i^{(m),t+1} = \sum_{j \in \mathcal{N}^{(m)}(i)} \exp\left(q_i^{(m),t} \cdot k_j^{(m),t}\right) v_j^{(m),t}, \quad \text{for } i \in \mathcal{I}^{(m)}, \tag{3}$$

where $q_i^{(m),t} = \mathcal{R}^{(m)}(\{q_j^{(m+1),t}\}_{j \in i^{(m,m+1)}})$, $k_i^{(m),t} = \mathcal{R}^{(m)}(\{k_j^{(m+1),t}\}_{j \in i^{(m,m+1)}})$, $v_i^{(m),t} = \mathcal{R}^{(m)}(\{v_j^{(m+1),t}\}_{j \in i^{(m,m+1)}})$, and $\mathcal{N}^{(m)}(i)$ is the set of the m-th level neighbors of $i \in \mathcal{T}^{(m)}$.

4.2.4 Decompose Operation

To mix the multilevel features $f_i^{(m),t+1}$ for $m = 1, \ldots, r$, we propose a decompose operation that reverses the reduce operation from level 1 to level $r-1$.

The decompose operator

$$D^{(m)}\colon f_i^{(m),t+1} \mapsto \left\{ f_j^{(m+1),t+\frac{1}{2}} \right\}_{j \in i^{(m,m+1)}},$$

maps the m-th level feature with index i to $(m+1)$-th level features associated to its child set $i^{(m,m+1)}$. The $(m+1)$-th level features $f_i^{(m+1),t+\frac{1}{2}}$ are further aggregated to $f_i^{(m+1),t+1}$, such that $f_i^{(m+1),t+1} \mathrel{+}= f_i^{(m+1),t+\frac{1}{2}}$ for $i \in \mathcal{T}^{(m+1)}$. In the current implementation, we use a simple linear layer such that $f_{i,s}^{(m+1),t+\frac{1}{2}} = D^{(m),s} T f_i^{(m),t+1}$, for $s = 0, 1, 2, 3$, with parameter matrices $D^{(m),s} \in \mathbb{R}^{C^{(m)} \times C^{(m+1)}}$.

4.3 Algorithm

Algorithm 1 One V-cycle of Hierarchical Attention

Input: $\mathcal{I}^{(r)}$, $\boldsymbol{f}_i^{(r),t}$ for $i \in \mathcal{I}^{(r)}$.

STEP 0: Get the $\boldsymbol{q}_i^{(r),t}$, $\boldsymbol{k}_i^{(r),t}$, $\boldsymbol{v}_i^{(r),t}$ for $i \in \mathcal{I}^{(r)}$.

STEP 1: For $m = r-1, \cdots, 1$, **Do** the reduce operations $\boldsymbol{q}_i^{(m),t} = \mathcal{R}^{(m)}(\{\boldsymbol{q}_j^{(m+1),t}\}_{j \in i^{(m,m+1)}})$ and also for $\boldsymbol{k}_i^{(m),t}$ and $\boldsymbol{v}_i^{(m),t}$, for any $i \in \mathcal{I}^{(m)}$.

STEP 2: For $m = r, \cdots, 1$, **Do** the local aggregation by equation 3.2 and equation 3.3 to get $\boldsymbol{f}_i^{(m),t+1}, m = 1, ..., r$ for any $i \in \mathcal{I}^{(m)}$.

STEP 3: For $m = 1, \cdots, r-1$, **Do** the decompose operations $\{\boldsymbol{f}_j^{(m+1),t+\frac{1}{2}}\}_{j \in i^{(m,m+1)}} = \mathcal{D}^{(m)}(\boldsymbol{f}_i^{(m),t+1})$, for any $i \in \mathcal{I}^{(m)}$; then $\boldsymbol{f}_i^{(m+1),t+1} += \boldsymbol{f}_i^{(m+1),t+\frac{1}{2}}$, for any $i \in \mathcal{I}^{(m+1)}$

Output: $\boldsymbol{f}_i^{(r),t+1}$ for any $i \in \mathcal{I}^{(r)}$.

The decoder is chosen as spetral convolution layer.

Empirical L^2 loss function is defined as

$$L^2(\{(a_j, u_j)\}_{j=1}^N; \theta) := \frac{1}{N} \sum_{i=1}^{N} \frac{\|u_j - \mathcal{N}(a_j; \theta)\|_2^2}{\|u_j\|_2^2},$$

where $\|\cdot\|_2$ is the canonical l^2 vector norm. For any $\xi \in \mathbb{Z}_n^2 := \{\xi \in \mathbb{Z}^2 | -n/2 + 1 \leq \xi_j \leq n/2, j = 1, 2\}$, the normalized discrete Fourier transform (DFT) coefficients of f writes

$$F(f)(\xi) := \frac{1}{\sqrt{n}} \sum_{x \in G^2} f(x)\, e^{-2\pi i x \cdot \xi}.$$

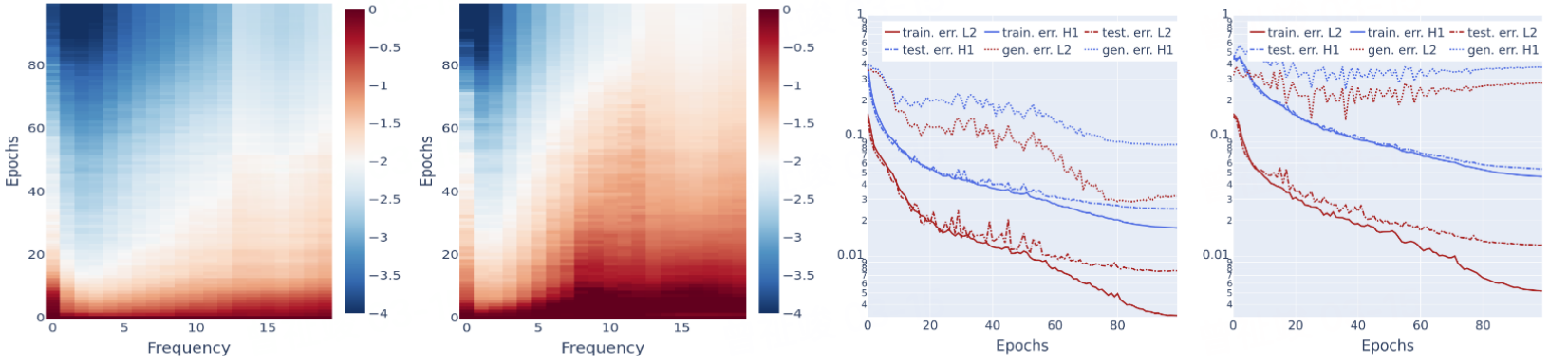
The empirical H^1 loss function is given by,

$$L^H(\{(a_j, u_j)\}_{j=1}^N; \theta) := \frac{1}{N} \sum_{i=1}^{N} \frac{\|u_j - \mathcal{N}(a_j; \theta)\|_h}{\|u_j\|_h},$$

where $\|u\|_h := \sqrt{\sum_{\xi \in \mathbb{Z}_n^2} |\xi|^2 |F(u)(\xi)|^2}$

Model	Runtime (s)	Darcy smooth		Darcy rough		Multiscale trigonometric	
		L^2	H^1	L^2	H^1	L^2	H^1
FNO2D	7.278	0.620	3.883	1.646 ±0.021	11.955±0.088	1.794	12.605
FNO2D-48	8.062	0.619	2.620	1.220 ±0.018	5.138 ±0.093	1.565	11.093
FNO2D-96	10.969	0.575	2.437	1.216 ±0.024	5.140 ±0.281	1.518	10.106
MWT	19.715	—	—	1.138 ±0.010	4.107 ±0.008	1.021	7.245
GT	38.219	0.945	3.365	1.790 ±0.012	6.269 ±0.418	1.052	8.207
SWIN	41.417	—	—	1.622 ±0.047	6.796 ±0.359	1.489	13.385
HT-NET	33.375	**0.291**	**0.815**	**0.571±0.001**	**1.371±0.001**	**0.603**	**2.633**

— MWT (Gupta et al., 2021) only supports resolution with powers of two.



(a) (b) (c) (d)

Figure 4.2: In (a) HT-Net trained with H^1 loss, and (b) HT-Net trained with L^2 loss, we show the evolution of errors with x-axis for frequency, y-axis for training epochs, and colorbar for the magnitude of L^2 error on each frequency in $\log_{10}$ scale, the error for each frequency is normalized frequency-wise by the error at epoch 0. The loss curves with training, testing, and generalization errors are shown in (c) for HT-Net trained with H^1 loss, and in (d) for HT-Net trained with L^2 loss.

5 Transformer For BVP

Mathematical Formulation of EIT: Identify the unknown inclusion D buried in Ω or equivalently $p = \mathcal{I}^D(x)$

Forward Model of EIT:

$$\nabla \cdot (\sigma \nabla \boldsymbol{u}) = 0 \text{ in } \Omega, \quad \sigma = \sigma_1 \text{ in } D, \sigma = \sigma_2 \text{ in } D^c$$

Current: $g = \sigma \nabla u \cdot \boldsymbol{n}|_{\partial\Omega}$ Neumann B.C

Voltages: $f = u|_{\partial\Omega}$ Dirichlet B.C

NtD Mapping

$$\Lambda_\sigma : g = \sigma \nabla u \cdot \boldsymbol{n}|_{\partial\Omega} \to f = u|_{\partial\Omega}$$

Forward Model $\mathcal{F}: \mathbb{P} \to \mathbb{M}$

$$\boldsymbol{m} = \mathcal{F}(\boldsymbol{p})$$

where $\boldsymbol{m} \in \mathbb{M}$ is the measurement of $\partial\Omega$,$\boldsymbol{p} \in \mathbb{P}$ is the coefficient to be recovered.

Inverse problem $\mathcal{F}^{(-1)}: \mathbb{M} \to \mathbb{P}$, $\boldsymbol{p} = \mathcal{F}^{(-1)}(\boldsymbol{m})$,essentially

$$\Lambda_\sigma \to \sigma$$

5.1 Finite Number of Data Pair

One measurement is a single data pair (g,f), and all the measurements form:

$$\mathbb{M} := H^{-1/2}(\partial\Omega) \times \Lambda_\sigma H^{-1/2}(\partial\Omega) = \{(g, f) : g \in H^{-1/2}(\partial\Omega),\ f = \Lambda_\sigma g\}$$

Infinite data pair: $\mathbb{M} = \{(g, f) : g \in H^{-1/2}(\partial\Omega),\ f = \Lambda_\sigma g\}$,

Finite data pair : $\mathbb{M}_L = \text{span}\{\{(g_l, f_l)\}_{l=1}^{L} : g_l \in H^{-1/2}(\partial\Omega),\ f_l = \Lambda_\sigma g_l\}$

Operator

$$\mathcal{F}_L^{-1} : \mathbb{M}_L \to \mathbb{P}$$

may not be well-defined.

5.2 Direct Sampling Method

The index function:

$$\mathcal{I}^D(x) \approx \mathcal{I}_1^D(x) := \frac{\boldsymbol{d}(x) \cdot \nabla \phi(x)}{|f - \Lambda_{\sigma_0} g|_{L^2(\partial\Omega)} |\eta_x|_Y}.$$

$$f - \Lambda_{\sigma_0} g \to \phi \to \boldsymbol{d} \to \eta_x$$

1. $f - \Lambda_{\sigma_0} g = (\Lambda_\sigma - \Lambda_{\sigma_0}) g$:the difference of NtD mapping (known data)
2. ϕ is the harmonic extension of $f - \Lambda_{\sigma_0} g$ with noise ξ

$$-\Delta \phi = 0 \quad \text{in} \quad \Omega, \quad \boldsymbol{n} \cdot \nabla \phi = (f - \Lambda_{\sigma_0} g) + \xi \quad \text{on} \quad \partial\Omega, \quad \int_{\partial\Omega} \phi \mathrm{d}s = 0$$

3. $\boldsymbol{d}(x)$ probing direction : empirical choice $\boldsymbol{d}(x) = \frac{\nabla \phi}{\|\phi(x)\|}$
4. $-\Delta \eta_x = -\boldsymbol{d}(x) \cdot \nabla \delta_x$ in Ω, $\boldsymbol{n} \cdot \nabla \eta_x = 0$ on $\partial\Omega$, $\int_{\partial\Omega} \eta_x \mathrm{d}s = 0$ where $\{\eta_x\}$ restricted on boundary is equipped with $|\cdot|_Y$ semi-norm

5.3 Integral Form of DSM

$$\mathcal{I}_1^D(x) := \frac{\boldsymbol{d}(x) \cdot \nabla \phi(x)}{|f - \Lambda_{\sigma_0} g|_{L^2(\partial\Omega)} |\eta_x|_Y}$$

Heuristics: the global information of ϕ should be used to locate a point x

$$\hat{\mathcal{I}}_1^D(x) := \frac{\int_\Omega \boldsymbol{d}(x) \cdot \mathcal{K}(x,y) \nabla \phi(y) \mathrm{d}y}{|f f - \Lambda_{\sigma_0} g\|_{L^2(\partial\Omega)} |\eta_x|_Y},$$

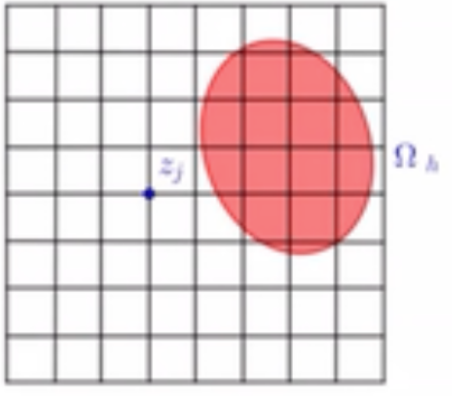
$$d(x) := \int_\Omega \mathcal{Q}(x,y) \nabla \phi(y) \mathrm{d}y,$$

$$|\eta_x|_Y^2 := (\mathcal{V}\eta_x, \eta_x)_{L^2(\partial\Omega)}.$$

Question: How to choose $\mathcal{K}(x,y), \mathcal{Q}(x,y), \mathcal{V}(x,y)$?

5.4 Matrix Form: Spatial Discretization

- Ω_h: a mesh of Ω with size h
- $\{z_j\}_{j=1}^{M}$ grid points
- $u \approx \boldsymbol{u}_h \in \mathbb{R}^M$
- ϕ_n: interpolation $\partial_{x_n}\phi$ at $\{z_j\}$, $n = 1, 2$



$$\mathcal{I}_1^D(x) := \frac{d(x) \cdot \nabla \phi(x)}{|f - \Lambda_{\sigma_0} g|_{L^2(\partial\Omega)} |\eta_x|_Y},$$

$$\hat{\mathcal{I}}_1^D(x) := \frac{\int_\Omega \boldsymbol{d}(x) \cdot \mathcal{K}(x,y) \nabla \phi(y) \mathrm{d}y}{|f - \Lambda_{\sigma_0} g|_{L^2(\partial\Omega)} |\eta_x|_Y},$$

$$d(x) := \int_\Omega \mathcal{Q}(x,y) \nabla \phi(y) \mathrm{d}y,$$

$$|\eta_x|_Y^2 := (\mathcal{V}\eta_x, \eta_x)_{L^2(\partial\Omega)},$$

$$\int_\Omega \mathcal{K}(z_i, y)\, \partial_{x_n} \phi(y) \mathrm{d}y \approx \sum_j \omega_j\, \mathcal{K}(z_i, z_j)\, \partial_{x_n} \phi(z_j) =: \boldsymbol{k}_i^\top\, \boldsymbol{\phi}_n.$$

$$d_n(z_i) \approx \sum_j \omega_j\, \mathcal{Q}(z_i, z_j)\, \partial_{x_n} \phi(z_j) =: \boldsymbol{q}_i^\top\, \boldsymbol{\phi}_n.$$

$$|\eta_{z_i}|_Y^2 \approx \bar{\boldsymbol{\eta}}_i^\top\, V \bar{\boldsymbol{\eta}}_i \approx \sum_n c_{n,i}\, \boldsymbol{\phi}_n^\top\, \boldsymbol{q}\, \boldsymbol{q}_i^\top\, \boldsymbol{\phi}_n = \sum_n \boldsymbol{\phi}_n^\top\, \boldsymbol{v}_{n,i}\, \boldsymbol{v}_{n,i}^\top\, \boldsymbol{\phi}_n$$

$$\hat{\mathcal{I}}_1^D(z_i) \approx C_{f,g} \left(\sum_n \boldsymbol{\phi}_n^\top\, \boldsymbol{v}_{n,i}\, \boldsymbol{v}_{n,i}^\top\, \boldsymbol{\phi}_n \right)^{-1/2} \sum_n \boldsymbol{\phi}_n^\top\, \boldsymbol{q}_i\, \boldsymbol{k}_i^\top\, \boldsymbol{\phi}_n,$$

where $C_{f,g} = \| f - \Lambda_{\sigma_0} g \|_{L^2(\partial\Omega)}^{-1}$ is a constant.

$$
\begin{aligned}
\mathcal{I}_1^D(x) &:= \frac{d(x) \cdot \nabla \phi(x)}{|f - \Lambda_{\sigma_0} g|_{L^2(\partial\Omega)} |\eta_x|_Y}, \\
\hat{\mathcal{I}}_1^D(x) &:= \frac{\int_\Omega \boldsymbol{d}(x) \cdot \mathcal{K}(x,y) \nabla \phi(y) \mathrm{d}y}{|f - \Lambda_{\sigma_0} g|_{L^2(\partial\Omega)} |\eta_x|_Y}, \\
d(x) &:= \int_\Omega \mathcal{Q}(x,y) \nabla \phi(y) \mathrm{d}y, \\
|\eta_x|_Y^2 &:= (\mathcal{V}\eta_x, \eta_x)_{L^2(\partial\Omega)},
\end{aligned}
$$

$$
[\hat{\mathcal{I}}_1^D(z_i)]_{i=1}^M \approx C_{f,g} \left(\phi_h W^Q * \phi_h W^K\right) / \left(\phi_h W^V * \phi_h W^V\right)^{1/2}
$$

How to choose $\mathcal{K}, \mathcal{Q}, \mathcal{V}$: Learn matrices W^Q, W^K, W^V using Transformer!

5.5 Generate Data

- Spatial discretization:Ω_h: a mesh of Ω, $\{z_j\}_{j=1}^M$: $\mathcal{M}$ grid points, $u \approx \boldsymbol{u}_h \in \mathbb{R}^M$
- Sampling of Coefficient:σ. Generate $N \gg 1$ samples of D with different shape and location

  $\boldsymbol{p}_h^k$: a discretization of $\mathcal{I}^D$ on Ω_h

  $\boldsymbol{m}_h^k$: a data function associated with $\mathcal{F}_L(\boldsymbol{p}_h^k)$
- Sampling of NtD mapping:For each sample of D, generate L input-output data pair

  $\{(g_l, f_l)_{l=1}^L\}$ to sample NtD mapping Λ_σ which are used to generated L channels in an input image.

5.6 Operator Learning Problems for EIT

Find a parametrization mapping $\mathcal{G}_\theta$ to approximate $\mathcal{F}^{-1}$ by minimizing :

$$\mathcal{J}(\theta) = \frac{1}{N}\Sigma_{k=1}^{N}\|\mathcal{G}_\theta(\boldsymbol{m}_h^k) - \boldsymbol{p}_h^k\|_{\mathbb{P}}^2$$

The parameters N, h, L will affect the approximation

1. h determines the resolution
2. N affect the representativity of training dataset
3. L decides how much of a finite portion of the infinite spectral information can be accessed

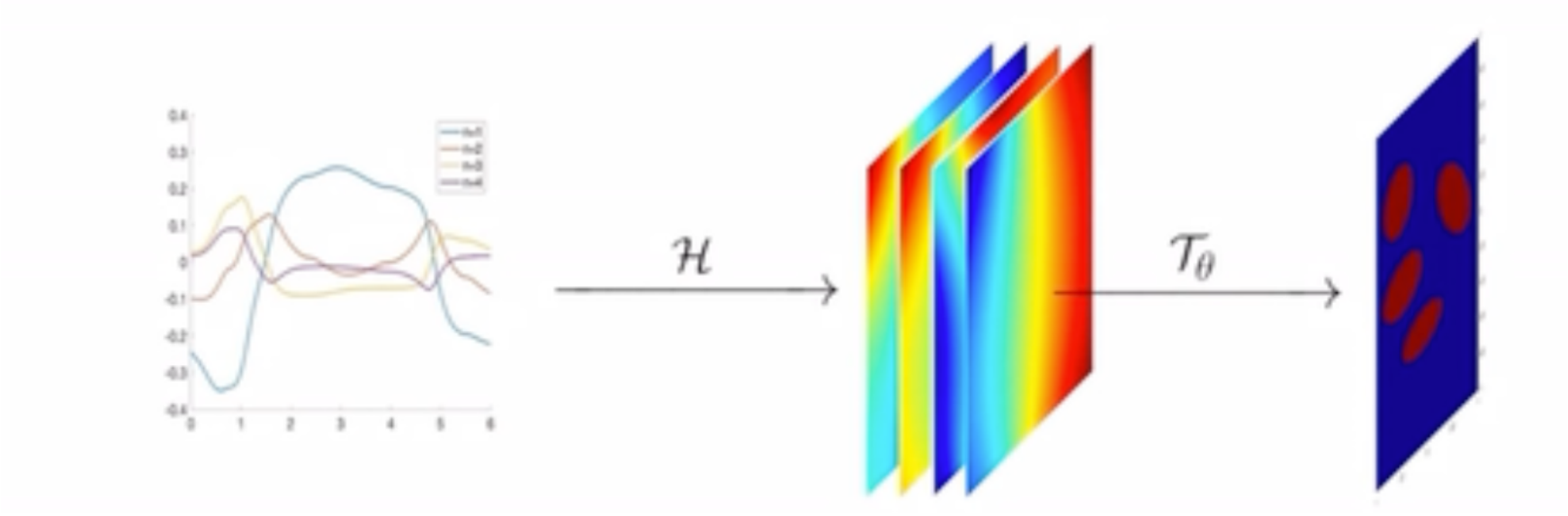
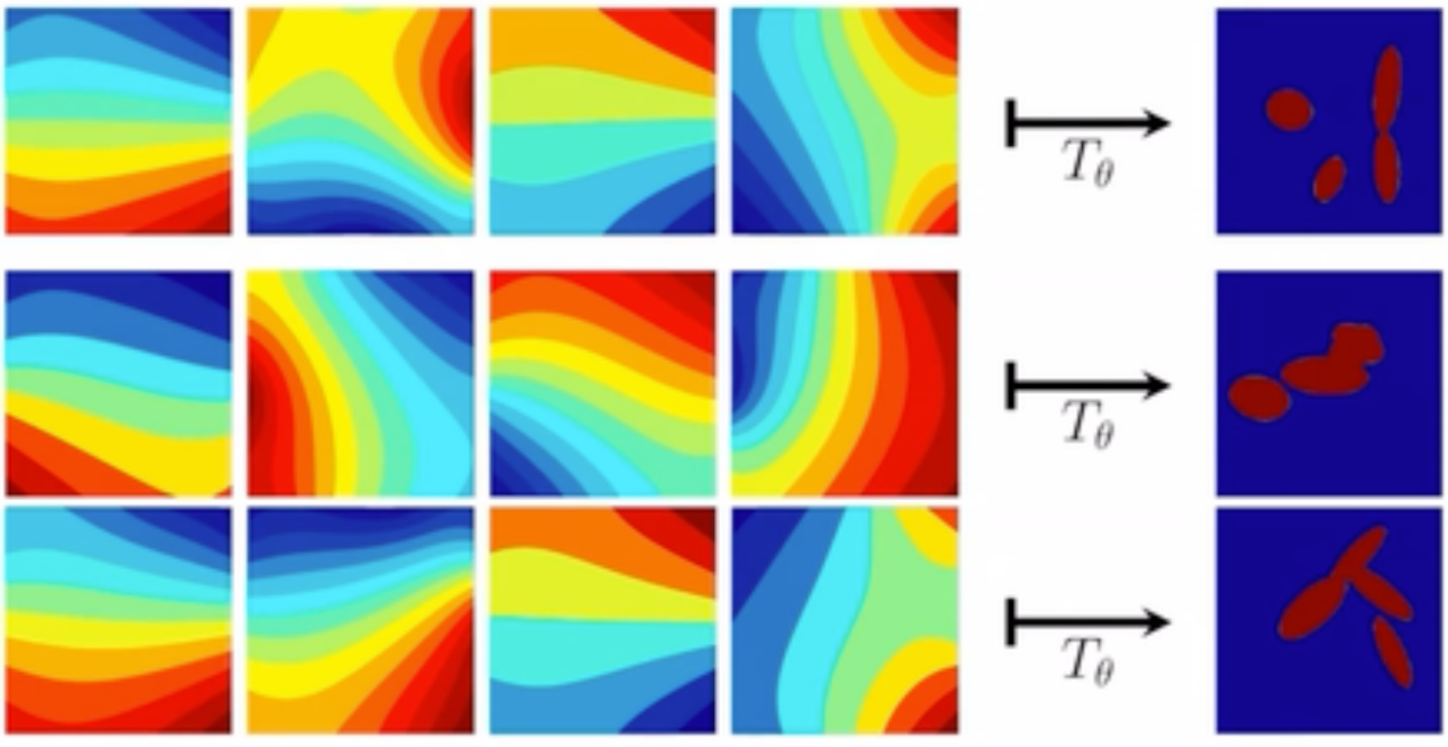
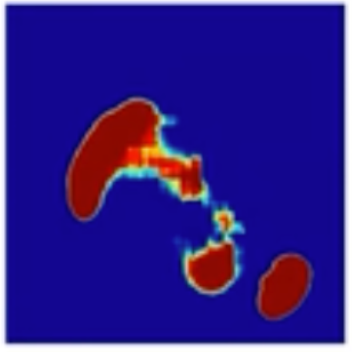


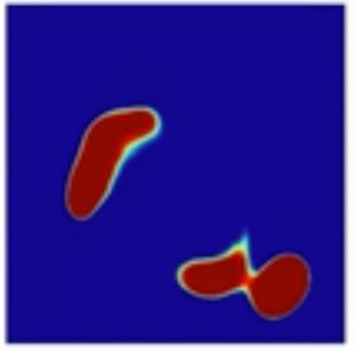
Image to Image



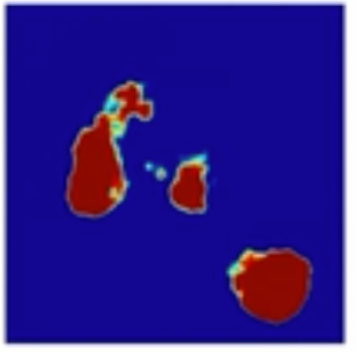
Learn a *single* parametrized operator T_θ that maps harmonic extension features to reconstruct the inclusions.



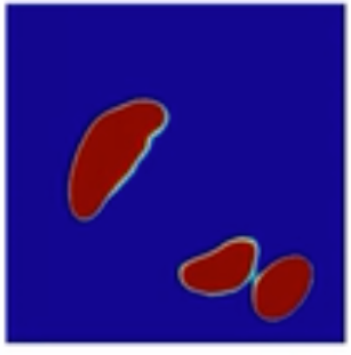
U-Net (7.7m, $L = 1$)



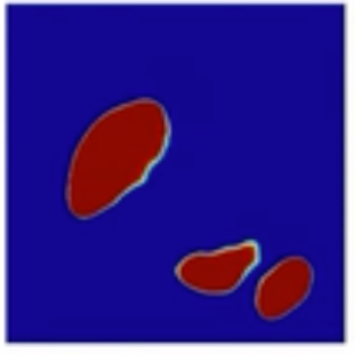
U-Net big (31m, $L = 3$)



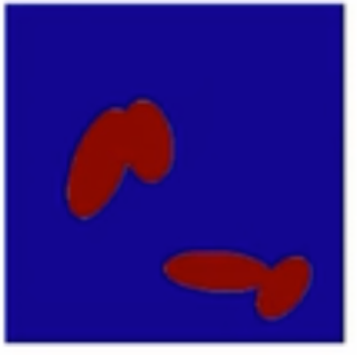
FNO (33m, $L = 1$)



UIT (11.4m, $L = 1$)



UIT (11.4m, $L = 3$)



Ground truth

6 PDEFormer

Consider 1D time-dependent PDEs on $(t, x) \in [0, 1] \times [0, 1]$ with periodic boundary condition

$$\mathcal{F}(u, c_1, \dots) = 0, u(0, x) = g(x)$$

where $c_i \in \mathbb{R}$ are real-valued coefficients, $g(x)$ is the initial condition. Assume that $\mathcal{F}$ has symbolic expression which involves differential and algebraic operations.

Goal: Construct a surrogate of the solution $(\mathcal{F}, g, c_1, \dots) \to u$ operator.

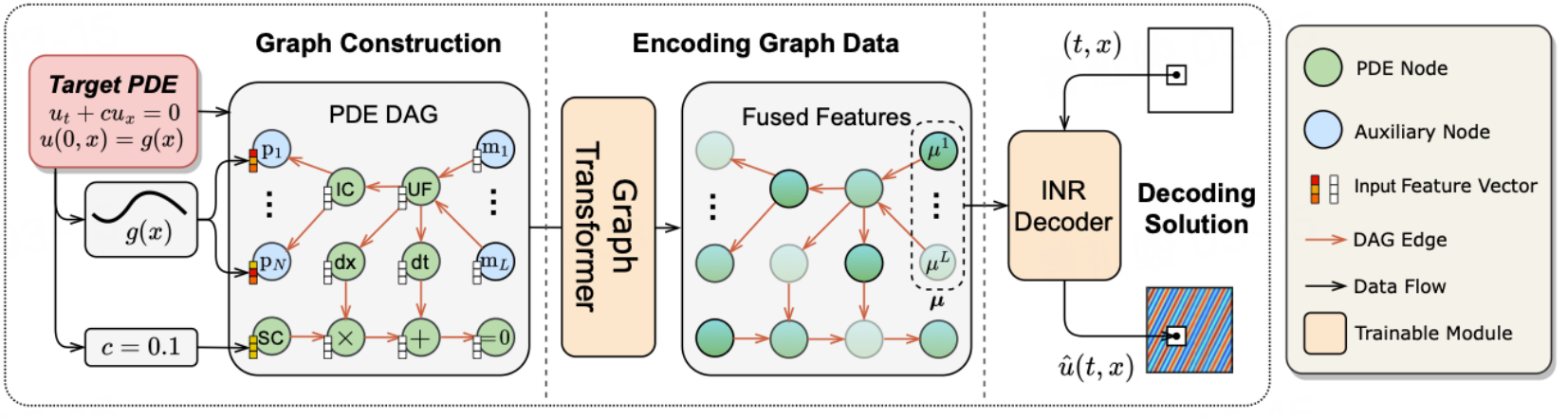


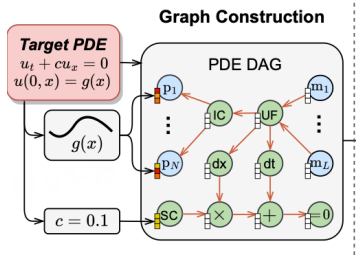
Figure 1: PDEformer architecture, taking $\mathcal{F}(u, c) = u_t + cu_x$ as the example.

6.1 Graph Construction

For $\mathcal{F}$, the symbolic information specifying the PDE, we use Computational Graph to represent.

Node: feature vector $\mathbb{R}^{d_f}$

1. Unknown field variable (UF): a node that represent unknown field u, we introduce $m_1 \ldots, m_L$ additional nodes that connected to UF, which is used to decode the prediction.
2. scalar coefficient (SC): c repeated d_f times
3. Initial condition (IC):$g(x)$ given at an equi-space with n_x grid point and divide into $N = n_x / d_f$ patches, yielding N vector $g_1, \ldots, g_N \in \mathbb{R}^{d_f}$, connect these node with IC node
4. Operation:node dx and dt for example.



6.2 Encoding Graph Data

We use graph-Transformer to encode symbolic and numeric information in the graph data into a latent code $\boldsymbol{\mu} = [\mu^1, \ldots, \mu^L] \in \mathbb{R}^{L \times d_e}$.

For $l = 1, \ldots, L$ we let $\mu^l \in \mathbb{R}^{d_e}$ be the embedding vector assigned to the node with type m_l in the output.

6.2.1 GNN

Let $G = (V, E)$ be a graph, feature vector of node v_i be x_i, GNNs aims to learn representation of nodes and graphs. Modern GNNs follow a learning schema that iteratively updates the representation of a node by aggregating representations of its first or higher-order neighbors.

$h_i^{(l)}$ denotes the representation of v_i at the $l-\text{th}$ layer and $h_i^{(0)} = x_i$. The AGGREGATE-COMBINE step as

$$a_i^{(l)} = \text{AGGREGATE}^{(l)}\left(\left\{h_j^{(l-1)} : j \in \mathcal{N}(v_i)\right\}\right), \quad h_i^{(l)} = \text{COMBINE}^{(l)}\left(h_i^{(l-1)}, a_i^{(l)}\right),$$

The AGGREGATE function is used to gather the information from neighbors. Common aggregation functions include MEAN, MAX, SUM, which are used in different architectures of GNNs. The goal of COMBINE function is to fuse the information from neighbors into the node representation.

for graph representation tasks, a READOUT function is designed to aggregate node features $h_i^{(L)}$ of the final iteration into h_G

$$h_G = \mathrm{READOUT}\left(\left\{h_i^{\{L\}} | v_i \in G\right\}\right)$$

6.2.2 Structural Encodings in Graphormer

Node centrality measures how important the a node is in the graph. We use the degree centrality as an additional signal to NN.

Assigns each node two real- valued embedding vectors according to its indegree and outdegree.

$$h_i^{(0)} = x_i + z^-_{\deg^-(v_i)} + z^+_{\deg^+(v_i)}$$

where $z^-, z^+ \in \mathbb{R}^d$ are learnable embedding vectors specified by indegree $\deg^-(v_i)$ and out degree $\deg^+(v_i)$

6.2.3 Spatial Encoding

For graphs, nodes are not arranged as a sequence. To encodee structral information of graph G, we consider a function $\phi(v_i, v_j): V \times V \to \mathbb{R}$ which measure the spatial relation between v_i, v_j if they connect. In GraphTransformer they use the shortest path as ϕ.

Modified Attention

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i, v_j)},$$

where $b_{\phi(v_i, v_j)}$ is a learnable scalar indexed by $\phi(v_i, v_j)$ and shared accross all layers.

6.2.4 Edge Encoding in Attention

To encode edge features into attention layers, we

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i, v_j)} + c_{ij}, \text{where } c_{ij} = \frac{1}{N} \sum_{n=1}^{N} x_{e_n} (w_n^E)^T$$

where x_{e_n} is the feature of the n-th edge e_n in $\mathrm{SP_{ij}} = (e_1, \ldots, e_N)$, $w_n^E \in \mathbb{R}^{d_E}$ is the n-th weight embedding and d_E is the dimensionality of edge feature.

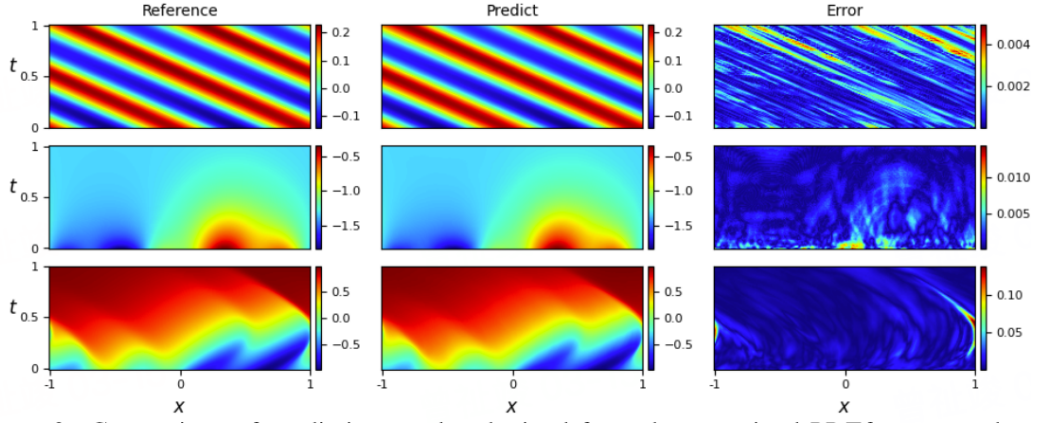
$$
\begin{aligned}
h'(l) &= \mathrm{MHA}(\mathrm{LN}(h^{(l-1)})) + h^{(l-1)} \\
h^{(l)} &= \mathrm{FFN}(\mathrm{LN}(h'(l))) + h'(l)
\end{aligned}
$$

6.3 Decoder

To obtain a mesh-free prediction, we employ implicit neural representation that takes (t, x) as input and produces $\hat{u}(t, x)$ according to $\boldsymbol{\mu}$.

Table 1: Test relative L^2 error on PDEBench. Format the first and second best outcomes in bold and underline respectively.

Model	Burgers			Adection		Reaction-Diffusion
	$\nu = 0.1$	$\nu = 0.01$	$\nu = 0.001$	$\beta = 0.1$	$\beta = 1$	$\nu = 1,\ \rho = 1$
U-Net (Ronneberger et al., 2015)	0.1627	0.2253	0.2431	0.0873	0.2655	0.0126
DeepONet (Lu et al., 2021)	0.0699	0.1791	0.2010	0.0186	0.0187	<u>0.0015</u>
FNO (Li et al., 2021)	0.0155	0.0445	<u>0.0700</u>	<u>0.0089</u>	<u>0.0097</u>	0.0018
PDEformer (Ours)	<u>0.0103</u>	<u>0.0309</u>	0.0921 (OoD)	0.0119	0.4000 (OoD)	0.7399 (OoD)
PDEformer-FT (Ours)	**0.0046**	**0.0146**	**0.0295**	**0.0043**	**0.0075**	**0.0009**



7 GNOT:General Neural Operator Transformer

Consider PDEs in the domain $\Omega \subset \mathbb{R}^d$ and the function space $\mathcal{H}$ over Ω, including boundary shapes and source functions.

Goal: learning an operator $\mathcal{G}$ from the input function space $\mathcal{A}$ to the solution space $\mathcal{H}$, i.e., $\mathcal{G}: \mathcal{A} \to \mathcal{H}$.

Input function space: $\mathcal{A}$ could contain multiple different types, like boundary shapes, source functions distributed over Ω, and vector parameters of the systems.

More formally, $\mathcal{A}$ could be represented as $\mathcal{A} = \mathcal{H} \times \cdots \times \mathcal{H} \times \mathbb{R}^p$. For $\forall a = (a^1(\cdot), \ldots, a^m(\cdot), \theta) \in \mathcal{A}$, $a^j(\cdot) \in \mathcal{H}$ represents boundary shapes and source functions, and $\theta \in \mathbb{R}^p$ represents parameters of the system, and $\mathcal{G}(a) = u \in \mathcal{H}$ is the solution function over Ω.

Dataset: $\mathcal{D} = \{(a_k, u_k)\}_{1 \leqslant k \leqslant D}$, where $u_k = \mathcal{G}(a_k)$. We discretize the input functions and the solution function on irregular discretized meshes over the domain Ω. For an input function a_k, we discretize it on the mesh $\{x_i^j \in \Omega\}_{1 \leqslant i \leqslant N_j}^{1 \leqslant j \leqslant m}$ and the discretized a_k^j is $\{(x_i^j, a_k^{i,j})\}_{1 \leqslant i \leqslant N_j}$, where $a_k^{i,j} = a_k^j(x_i^j)$. In this way, we use $\mathcal{A}_k = \{(x_i^j, a_k^{i,j})\}_{1 \leqslant i \leqslant N_j}^{1 \leqslant j \leqslant m} \cup \theta_k$ to represent the input functions a_k.

Solution function: u_k is discretized on mesh $\{y_i \in \Omega\}_{1 \leqslant i \leqslant N'}$ and become $\{(y_i, u_k^i)\}_{1 \leqslant i \leqslant N'}$, here $u_k^i = u_k(y_i)$.

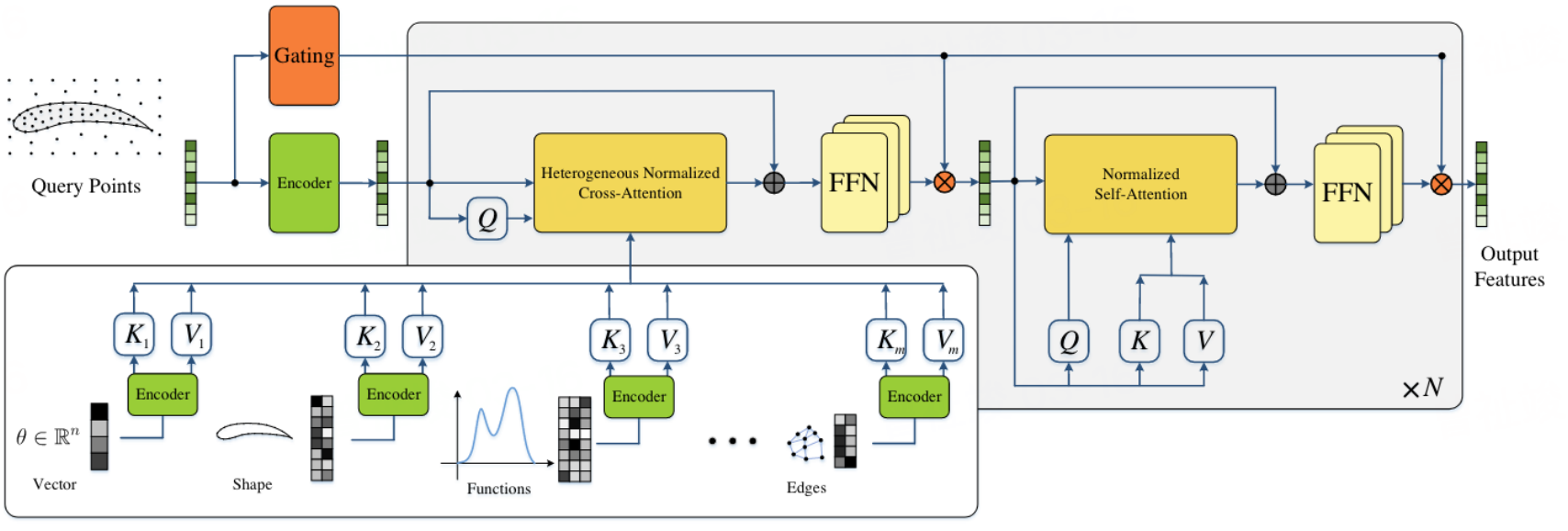
Operator $\mathcal{G}$: Use a parameterized neural network $\tilde{\mathcal{G}}_w$, which receives the input $\mathcal{A}_k (k = 1, ..., D)$ and outputs $\tilde{\mathcal{G}}_w(\mathcal{A}_k) = \{\tilde{u}_k^i\}_{1 \leqslant i \leqslant N'}$ to approximate u_k. Our goal is to minimize the mean squared error(MSE) loss between the prediction and data as

$$\min_{w \in W} \frac{1}{D} \sum_{k=1}^{D} \frac{1}{N'} \|\tilde{\mathcal{G}}_w(\mathcal{A}_k) - \{u_k^i\}_{1 \leqslant i \leqslant N'}\|_2^2, \tag{4}$$

where w is a set of the network parameters and W is the parameter space.

7.1 Architecture Overview

1. General input encoder: embed different input functions with MLPs
2. Attention block (self/cross) to modelling the latent operator
3. Geometric gating mechanism to tackle multiscale properties



7.2 General Input Encoding

Input of model :

Query Point:$\{x_i^q\}_{1\leqslant i\leqslant N_q}$ map into query embedding $X \in \mathbb{R}^{N_q \cdot n_e}$

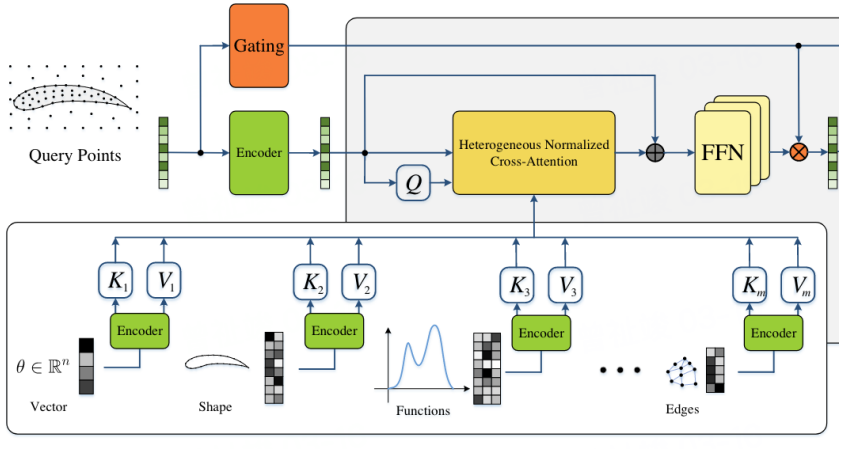
Parameter vector $\theta \in \mathbb{R}^p$ map into $Y = f_\omega(\theta) \in \mathbb{R}^{1\times n_e}$

Boundary shape $\{x_i\}_{1\leqslant i\leqslant N}$, map into $Y = (f_w(x_i))_{1\leqslant i\leqslant N} \in \mathbb{R}^{N\cdot n_e}$

Domain distributed function $\{(x_i, a_i)\}_{1\leqslant i\leqslant N}$ map into $Y = (f_w(x_i, a_i))_{1\leqslant i\leqslant N} \in \mathbb{R}^{Nn_e}$

Mesh points and information $\{(x_i, z_i)\}_{1\leqslant i\leqslant N}$ map into $Y = (f_w(x_i, z_i))_{1\leqslant i\leqslant N}$

Edge information $\{(x_i^{\text{src}}, x_i^{\text{dst}}, e_i)\}_{1\leqslant i\leqslant N}$ map into $Y = (f_w(x_i^{\text{src}}, x_i^{\text{dst}}, e_i))_{1\leqslant i\leqslant N}$



7.3 Heterogeneous Normalized Attention Block

Here we have features of query points X and conditional embeddings $\{Y_l\}_{l=1}^{L}$. We need to design self attention for X and cross attention for X and Y_l.

Suppose we have three sequences called queries $\{\boldsymbol{q}_i\}_{1\leqslant i\leqslant N}$, keys$\{\boldsymbol{k}_i\}_{1\leqslant i\leqslant M}$ and values $\{\boldsymbol{v}_i\}_{1\leqslant i\leqslant M}$. The attention is computed as follows,

$$\boldsymbol{z}_t = \sum_i \frac{\exp\left(\boldsymbol{q}_t \cdot \boldsymbol{k}_i / \tau\right)}{\sum_j \exp\left(\boldsymbol{q}_t \cdot \boldsymbol{k}_j / \tau\right)} \boldsymbol{v}_i, \tag{5}$$

where τ is a hyperparameter.

For self-attention models, $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$ are obtained by applying a linear transformation to input sequence $X = (\boldsymbol{x}_i)_{1\leqslant i\leqslant N}$, i. e, $\boldsymbol{q}_i = W_q\,\boldsymbol{x}_i$, $\boldsymbol{k}_i = W_k\,\boldsymbol{x}_i$, $\boldsymbol{v}_i = W_v\,\boldsymbol{x}_i$. For cross attention models, $\boldsymbol{q}$ comes from the query sequence X while keys and values come from another sequence $Y = (\boldsymbol{y}_i)_{1\leqslant i\leqslant M}$, i.e, $\boldsymbol{q}_i = W_q\,\boldsymbol{x}_i$, $\boldsymbol{k}_i = W_k\,\boldsymbol{y}_i$, $\boldsymbol{v}_i = W_v\,\boldsymbol{y}_i$.

However, the computational cost of the attention is $O(N^2\,n_e)$ for self attention and $O(N\,M\,n_e)$ for cross attention where n_e is the dimension of embedding.

We first normalize these sequences respectively,

$$\tilde{\boldsymbol{q}}_i = \mathrm{Softmax}(\boldsymbol{q}_i) = \left(\frac{e^{q_{ij}}}{\sum_j e^{q_{ij}}} \right)_{j=1,\dots n_e}, \tag{6}$$

$$\tilde{\boldsymbol{k}}_i = \mathrm{Softmax}(\boldsymbol{k}_i) = \left(\frac{e^{k_{ij}}}{\sum_j e^{k_{ij}}} \right)_{j=1,\dots n_e}. \tag{7}$$

Then we compute the attention output without softmax using the following equation,

$$\boldsymbol{z}_t = \sum_i \frac{\tilde{\boldsymbol{q}}_t \cdot \tilde{\boldsymbol{k}}_i}{\sum_j \tilde{\boldsymbol{q}}_t \cdot \tilde{\boldsymbol{k}}_j} \cdot \boldsymbol{v}_i. \tag{8}$$

We denote $\alpha_t = (\sum_j \tilde{\boldsymbol{q}}_t \cdot \tilde{\boldsymbol{k}}_j)^{-1}$ and the efficient attention could be represented by,

$$\boldsymbol{z}_t = \sum_i \alpha_t \, (\tilde{\boldsymbol{q}}_t \cdot \tilde{\boldsymbol{k}}_i) \cdot \boldsymbol{v}_i = \alpha_t \, \tilde{\boldsymbol{q}}_t \cdot \left(\sum_i \tilde{\boldsymbol{k}}_i \otimes \boldsymbol{v}_i \right). \tag{9}$$

We could compute $\sum_i \tilde{\boldsymbol{k}}_i \otimes \boldsymbol{v}_i$ first with a cost $O(M\, n_e^2)$ and then compute its multiplication with $\boldsymbol{q}$ with a cost $O(N\, n_e^2)$. The total cost is $O((M+N)\, n_e^2)$ which is linear with respect to the sequence length.

To fuse the query embeddings with multiple conditional embeddings, we design cross attention with above mechanism:

Specifically, suppose we have L conditional embeddings $\{Y_l \in \mathbb{R}^{N_l \times n_e}\}_{1 \leqslant l \leqslant L}$ encoding the input functions and extra information. We first compute the queries $Q = (\boldsymbol{q}_i) = XW_q$, keys $K_l = (\boldsymbol{k}_i^l) = YW_k$ and values $V_l = (\boldsymbol{v}_i^l) = YW_v$,and then normalize every $\boldsymbol{q}_i$ and $\boldsymbol{k}_i$ to be $\tilde{\boldsymbol{q}}_i$ and $\tilde{\boldsymbol{k}}_i$. Then we compute the cross-attention as follows,

$$\begin{aligned} \boldsymbol{z}_t &= \tilde{\boldsymbol{q}}_t + \frac{1}{L}\sum_{l=1}^{L}\sum_{i_l=1}^{N_l} \alpha_t^l \left(\tilde{\boldsymbol{q}}_t \cdot \tilde{\boldsymbol{k}}_{i_l}\right) \boldsymbol{v}_{i_l}, \qquad (10) \\ &= \tilde{\boldsymbol{q}}_t + \frac{1}{L}\sum_{l=1}^{L} \alpha_t^l \, \tilde{\boldsymbol{q}}_t \cdot \left(\sum_{i_l=1}^{N_l} \tilde{\boldsymbol{k}}_{i_l} \otimes \boldsymbol{v}_{i_l}\right). \qquad (11) \end{aligned}$$

where $\alpha_t^l = \frac{1}{\sum_{j=1}^{N_l} \tilde{\boldsymbol{q}}_t \cdot \tilde{\boldsymbol{k}}_j}$ is the normalization cofficient.

After applying such a cross-attention layer, we impose the self-attention layer for query features, i.e,

$$\boldsymbol{z}_t' = \sum_i \alpha_t \left(\tilde{\boldsymbol{q}}_t \cdot \tilde{\boldsymbol{k}}_i\right) \cdot \boldsymbol{v}_i, \tag{12}$$

where all of $\boldsymbol{q}$, $\boldsymbol{k}$ and $\boldsymbol{v}$ are computed with the embedding $\boldsymbol{z}_t$ as

$$\boldsymbol{q}_t = W_q\, \hat{\boldsymbol{z}}_t, \boldsymbol{k}_t = W_k\, \hat{\boldsymbol{z}}_t, \boldsymbol{v}_t = W_v\, \hat{\boldsymbol{z}}_t. \tag{13}$$

We tile multiple layers and multiple heads similar to other transformer models. The embedding $\boldsymbol{z}_t$ and $\boldsymbol{z}_t'$ are divided into H heads as $\boldsymbol{z}_t = \mathrm{Concat}(\boldsymbol{z}_t^i)_{i=1}^H$ and $\boldsymbol{z}_t' = \mathrm{Concat}(\boldsymbol{z'}_t^i)_{i=1}^H$

7.4 Geometric Gating Mechanism

To handle multi-scale problems, we introduce our geometric gating mechanism based on mixture-of-experts (MoE): a geometric gating network that inputs the coordinates of the query points and outputs unnormalized scores $G_i(x)$ for averaging these expert networks.

In each layer of our model, we use K subnetworks for the MLP denoted by $E_i(\cdot)$. The update of $\boldsymbol{z}_t$ and $\boldsymbol{z}_t'$ in the feedforward laye is replaced by the following equation when we have multiple expert networks as

$$\boldsymbol{z}_t \leftarrow \boldsymbol{z}_t + \sum_{i=1}^{K} p_i(x_t) \cdot E_i(\boldsymbol{z}_t). \tag{14}$$

The weights for averaging the expert networks are computed as

$$p_i(x_t) = \frac{\exp\left(G_i(x_t)\right)}{\sum_{i=1}^{K} \exp\left(G_i(x_t)\right)}, \tag{15}$$

where the gating network $G(\cdot)\colon \mathbb{R}^d \to \mathbb{R}^K$ takes the geometric coordinates of query points x_t as inputs. The normalized outputs $p_i(x_t)$ are the weights for averaging these experts.

7.5 Experiment

"A", "B", and "C" represent the problem has irregular mesh, has multiple input functions, and is multi-scale

Dataset	Type		MIONet	FNO(-interp)	GK-Transformer	Geo-FNO	OFormer	Ours
	Challenge	Subset						
Darcy2d	-	-	5.45e-2	1.09e-2	**8.40e-3**	1.09e-2	1.24e-2	1.05e-2
NS2d	-	part	–	1.56e-1	1.40e-1	1.56e-1	1.71e-1	**1.38e-1**
	-	full	–	8.20e-2	7.92e-2	8.20e-2	6.46e-2	**4.43e-2**
Elasticity	A	-	9.65e-2	5.08e-2	2.01e-2	2.20e-2	1.83e-2	**8.65e-3**
NS2d-c	A, C	u	2.74e-2	6.56e-2	1.52e-2	1.41e-2	2.33e-2	**6.73e-3**
		v	5.51e-2	1.15e-1	3.15e-2	2.98e-2	4.83e-2	**1.55e-2**
		p	2.74e-2	1.11e-2	1.59e-2	1.62e-2	2.43e-2	**7.41e-3**
NACA	A, C	-	1.32e-1	4.21e-2	1.61e-2	1.38e-2	1.83e-2	**7.57e-3**
Inductor2d	A, C	A_z	3.10e-2	–	2.56e-1	–	2.23e-2	**1.21e-2**
		B_x	3.49e-2	–	3.06e-2	–	2.83e-2	**1.92e-2**
		B_y	6.73e-2	–	4.45e-2	–	4.28e-2	**3.62e-2**
Heat	A, B, C	part	1.74e-1	–	–	–	–	**4.13e-2**
		full	1.45e-1	–	–	–	–	**2.56e-2**
Heatsink	A, B, C	T	4.67e-1	–	–	–	–	**2.53e-1**
		u	3.52e-1	–	–	–	–	**1.42e-1**
		v	3.23e-1	–	–	–	–	**1.81e-1**
		w	3.71e-1	–	–	–	–	**1.88e-1**

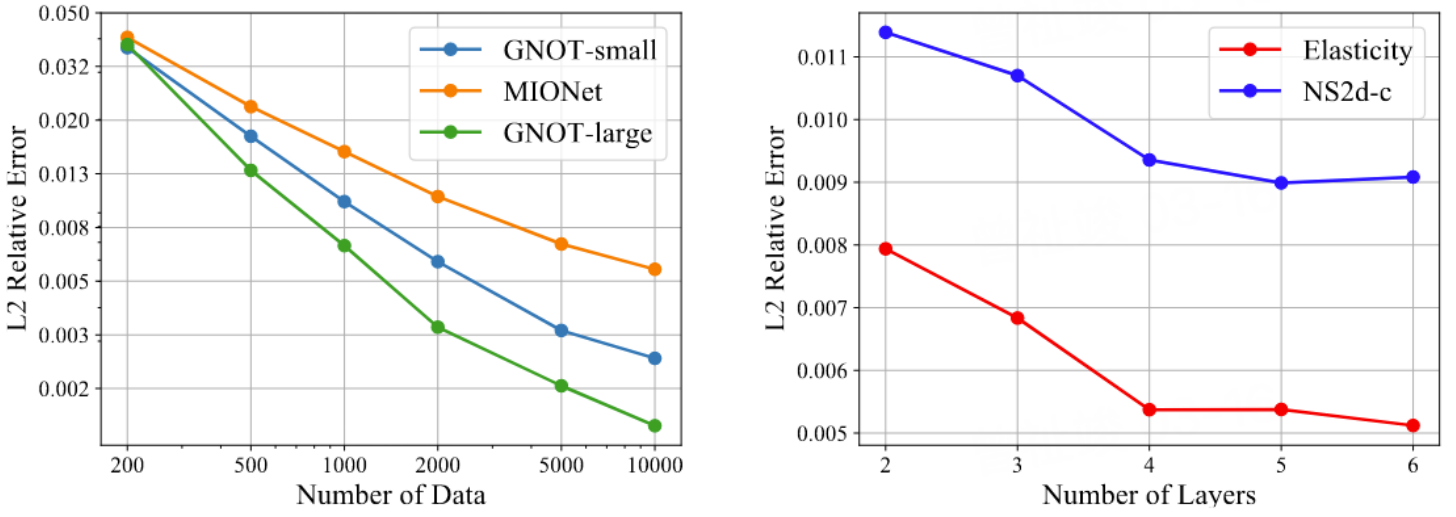


Figure 3. Results of scaling experiments for different dataset sizes (left) and different numbers of layers (right).

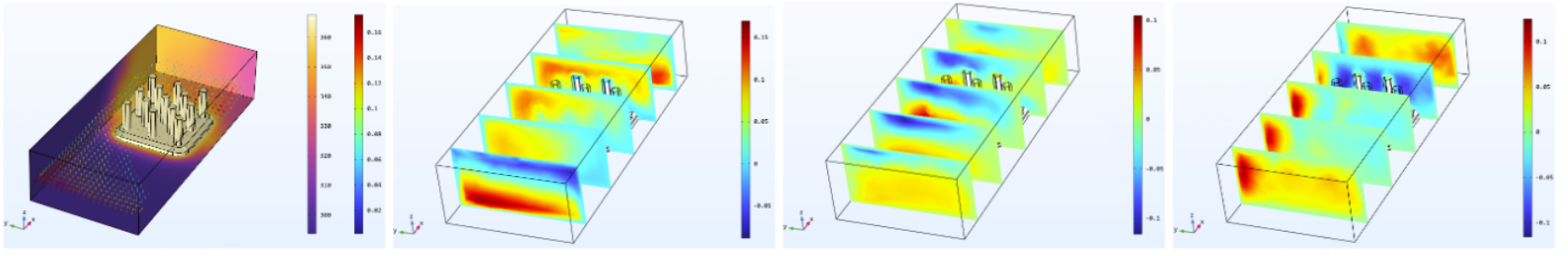


Figure 10. Visualization of T, u, v, w of Heatsink dataset.

8 Transolver: Transformer Solver for PDEs on General Geometry

Consider PDEs defined on input domain $\Omega \subset \mathbb{R}^{C_{\mathbf{g}}}$, where $C_{\mathbf{g}}$ denotes the dimension of input space. For numerical calculation, Ω is firstly discretized into a finite set of N mesh points $\mathbf{g} \in \mathbb{R}^{N \times C_{\mathbf{g}}}$. The task is to estimate target physical quantities based on input geometrics $\mathbf{g}$ and quantities $\mathbf{u} \in \mathbb{R}^{N \times C_{\mathbf{u}}}$ observed on $\mathbf{g}$. Here $\mathbf{u}$ is optional in some PDE-governed tasks.

Physics-Aware Tokens: Given a mesh set $\mathbf{g} = \{\mathbf{g}_i\}_{i=1}^{N}$ with the coordinate information of N mesh points and observed quantities $\mathbf{u}$, we firstly embed them into deep features $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^{N}$ by a linear layer contains C channels,$\mathbf{x}_i \in \mathbb{R}^{1 \times C}$. Then we propose a bottom-up paradigm, that ascribes each mesh point $\mathbf{g}_i$ to M potential slices based on its learned feature $\mathbf{x}_i$, which is formalized as follows:

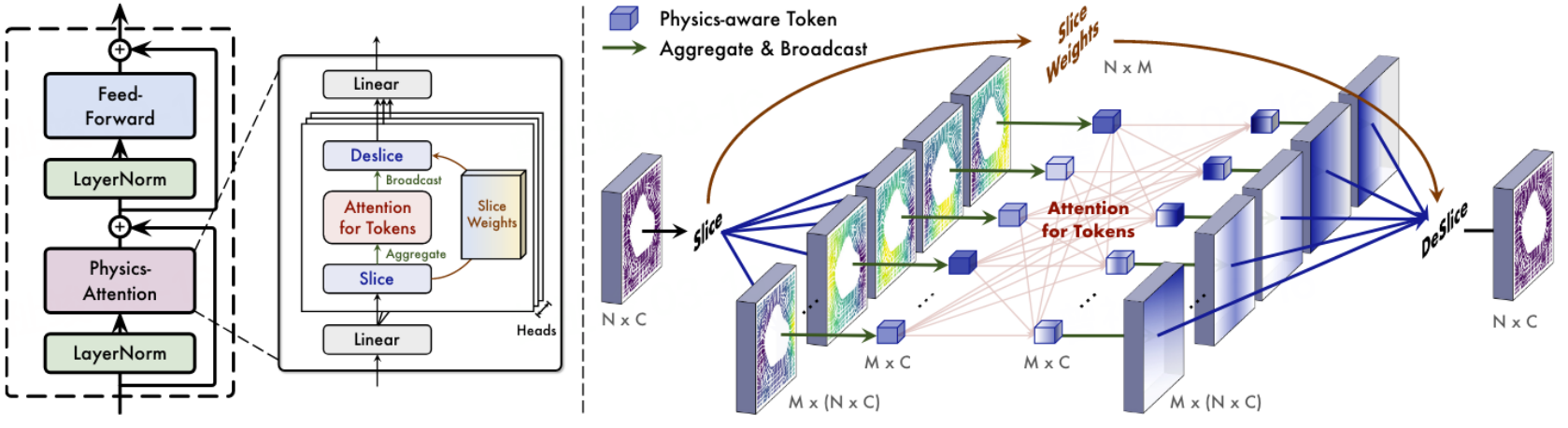
$$\begin{aligned} \{\mathbf{w}_i\}_{i=1}^{N} &= \{\mathrm{Softmax}(\mathrm{Linear}(\mathbf{x}_i))\}_{i=1}^{N} \\ \mathbf{s}_j &= \{\mathbf{w}_{i,j}\mathbf{x}_i\}_{i=1}^{N}, \end{aligned} \tag{16}$$

where $\mathrm{Linear}()$ projects C channels into M weights and yields slice weights $\mathbf{w}_i \in \mathbb{R}^{1 \times M}$ after $\mathrm{Softmax}()$. Specifically, $\mathbf{w}_{i,j}$ represents the degree that the i-th mesh point belongs to the j-th slices with $\sum_{j=1}^{M} \mathbf{w}_{i,j} = 1$. $\mathbf{s}_j \in \mathbb{R}^{N \times C}$ represents the j-th slice feature, which is a weighted combination of N mesh point features $\mathbf{x}$.

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

$$\mathbf{z}_j = \frac{\sum_{i=1}^{N} \mathbf{s}_{j,i}}{\sum_{i=1}^{N} \mathbf{w}_{i,j}} = \frac{\sum_{i=1}^{N} \mathbf{w}_{i,j}\mathbf{x}_i}{\sum_{i=1}^{N} \mathbf{w}_{i,j}}, \tag{17}$$

where $\mathbf{z}_j \in \mathbb{R}^{1\times C}$.



8.1 Physics-Attention

After obtained physics-aware tokens $\mathbf{z} = \{\mathbf{z}_j\}_{j=1}^{M} \in \mathbb{R}^{M \times C}$, we employ the attention mechanism among encoded tokens to capture intricate correlations among different physical states, that is

$$\begin{aligned} \mathbf{q}, \mathbf{k}, \mathbf{v} &= \mathrm{Linear}(\mathbf{z}) \\ \mathbf{z}' &= \mathrm{Softmax}\left(\frac{\mathbf{q}\mathbf{k}^T}{\sqrt{C}}\right)\mathbf{v}, \end{aligned} \tag{18}$$

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

$$\mathbf{x}'_i = \sum_{j=1}^{M} \mathbf{w}_{i,j}\mathbf{z}'_j, \tag{19}$$

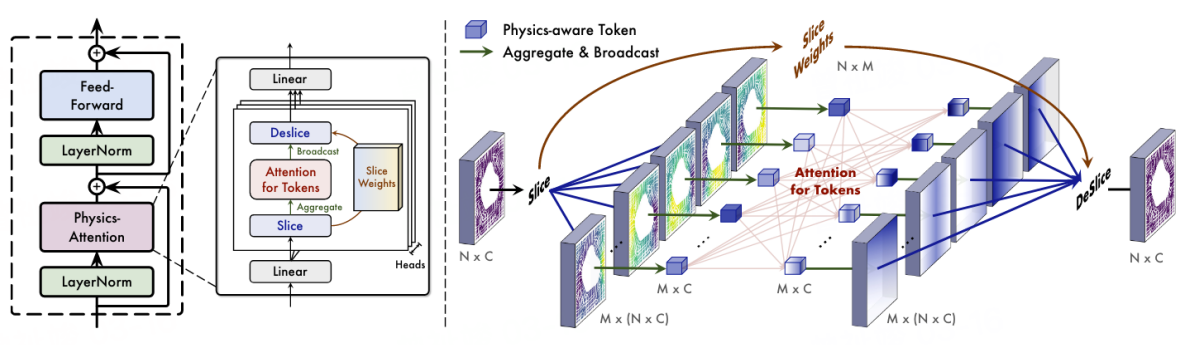
where $1 \leq i \leq N$ and each token $\mathbf{z}'_j$ is broadcasted to all mesh points during above calculation. For clarity, we summarize the above process as $\mathbf{x}' = \mathrm{Physics-Attn}(\mathbf{x})$, whose overall complexity is $\mathcal{O}\left(NMC + M^2C\right)$.

8.2 Overall Design

We propose the Transolver by replacing the attention mechanism with Physics-Attention. Suppose there are L layers, the l-th layer of Transolver can be formalized as follows:

$$\begin{aligned}\hat{\mathbf{x}}^l &= \text{Physics} - \text{Attn}(\text{LayerNorm}(\mathbf{x}^{l-1})) + \mathbf{x}^{l-1} \\ \mathbf{x}^l &= \text{FeedForward}(\text{LayerNorm}(\hat{\mathbf{x}}^l)) + \hat{\mathbf{x}}^l,\end{aligned} \tag{20}$$

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



8.3 Physics-Attention is equivalent to learnable integral on Ω

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

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

$$\mathcal{G}(\boldsymbol{u})(\mathbf{g}^*) = \int_{\Omega} \kappa(\mathbf{g}^*, \boldsymbol{\xi})\, \boldsymbol{u}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}, \tag{21}$$

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

$$\kappa(\mathbf{g}^*, \boldsymbol{\xi}) = \left(\int_{\Omega} \exp\left((\mathbf{W_q} \boldsymbol{u}(\boldsymbol{\xi}'))(\mathbf{W_k} \boldsymbol{u}(\boldsymbol{\xi}))^T \right) \mathrm{d}\boldsymbol{\xi}' \right)^{-1} \exp\left((\mathbf{W_q} \boldsymbol{u}(\mathbf{g}^*))(\mathbf{W_k} \boldsymbol{u}(\boldsymbol{\xi}))^T \right) \mathbf{W_v}, \tag{22}$$

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

证明. Suppose that there are N discretized mesh points $\{\mathbf{g}_1, \cdots, \mathbf{g}_N\}$, where $\mathbf{g}_i \in \Omega \subset \mathbb{R}^{C_\mathbf{g}}$. Approximating the inner-integral in Eq. (22) by Monte-Carlo, we have:

$$\int_\Omega \exp\left((\mathbf{W_q}\boldsymbol{u}(\boldsymbol{\xi}'))(\mathbf{W_k}\boldsymbol{u}(\boldsymbol{\xi}))^T\right)\mathrm{d}\boldsymbol{\xi}' \approx \frac{|\Omega|}{N}\sum_{i=1}^{N}\exp\left((\mathbf{W_q}\boldsymbol{u}(\mathbf{g}_i))(\mathbf{W_k}\boldsymbol{u}(\boldsymbol{\xi}))^T\right). \tag{23}$$

Applying the above equation to Eq. (21) and using the same approximation for the outer-integral, we have:

$$\mathcal{G}(\boldsymbol{u})(\mathbf{g}^*) \approx \sum_{i=1}^{N}\frac{\exp\left((\mathbf{W_q}\boldsymbol{u}(\mathbf{g}^*))(\mathbf{W_k}\boldsymbol{u}(\mathbf{g}_i))^T\right)\mathbf{W_v}\boldsymbol{u}(\mathbf{g}_i)}{\sum_{j=1}^{N}\exp\left((\mathbf{W_q}\boldsymbol{u}(\mathbf{g}_j))(\mathbf{W_k}\boldsymbol{u}(\mathbf{g}_i))^T\right)}, \tag{24}$$

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

引理 2. Suppose that Ω is a countable domain, the slice domain Ω_{s} is isomorphic to Ω.

定理 3. (Physics-Attention is equivalent to learnable integral on Ω) Given input function $\boldsymbol{u}: \Omega \to \mathbb{R}^C$ and a mesh point $\mathbf{g}^* \in \Omega$, Physics-Attention is to approximate the integral operator $\mathcal{G}$, which is defined as:

$$\mathcal{G}(\boldsymbol{u})(\mathbf{g}^*) = \int_{\Omega} \kappa(\mathbf{g}^*, \boldsymbol{\xi})\, \boldsymbol{u}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}, \tag{25}$$

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

证明. According to Lemma 2, we can obtain an isomorphic projection $\boldsymbol{g}$ between a countable input domain Ω and slice domain Ω_{s}. Suppose that the slice weight $w_{*,*}$: $\bar{\Omega} \times \bar{\Omega}_{\mathrm{s}} \to \mathbb{R}$ is smooth in both $\bar{\Omega}$ and $\bar{\Omega}_{\mathrm{s}}$, where the $\bar{\Omega}$ and $\bar{\Omega}_{\mathrm{s}}$ denote the continuation of Ω and Ω_{s} respectively, we can obtain $\boldsymbol{g}$ as a diffeomorphism projection.

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

$$\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s}}) = \left(\int_{\Omega} w_{\boldsymbol{\xi}, \boldsymbol{\xi}_{\mathrm{s}}}\, \boldsymbol{u}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \right) / \left(\int_{\Omega} w_{\boldsymbol{\xi}, \boldsymbol{\xi}_{\mathrm{s}}} \mathrm{d}\boldsymbol{\xi} \right), \tag{26}$$

which corresponds to the slice token definition in Eq. (17). □

证明. Based on the above assumptions and definitions, we have:

$$
\begin{aligned}
&\mathcal{G}(\boldsymbol{u})(\mathbf{g}) = \int_{\Omega} \kappa(\mathbf{g}, \boldsymbol{\xi})\, \boldsymbol{u}(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \\
&= \int_{\Omega_{\mathrm{s}}} \kappa_{\mathrm{ms}}(\mathbf{g}, \boldsymbol{\xi}_{\mathrm{s}})\, \boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s}}) \mathrm{d}\boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}}) \\
&= \int_{\Omega_{\mathrm{s}}} \kappa_{\mathrm{ms}}(\mathbf{g}, \boldsymbol{\xi}_{\mathrm{s}})\, \boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s}}) |\det\left(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}} \boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}})\right)| \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}}(\boldsymbol{\xi}_{\mathrm{s}}', \boldsymbol{\xi}_{\mathrm{s}}) \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}}(\boldsymbol{\xi}_{\mathrm{s}}) |\det\left(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}} \boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}})\right)| \mathrm{d}\boldsymbol{\xi}_{\mathrm{s}} \\
&= \int_{\Omega_{\mathrm{s}}} \underbrace{w_{\mathbf{g}, \boldsymbol{\xi}_{\mathrm{s}}'}}_{\text{DeSlice}} \int_{\Omega_{\mathrm{s}}} \underbrace{\kappa_{\mathrm{ss}}(\boldsymbol{\xi}_{\mathrm{s}}', \boldsymbol{\xi}_{\mathrm{s}})}_{\text{Attention among slice tokens}} \underbrace{\boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s}})}_{\text{Slice token}} |\det\left(\nabla_{\boldsymbol{\xi}_{\mathrm{s}}} \boldsymbol{g}^{-1}(\boldsymbol{\xi}_{\mathrm{s}})\right)| \mathrm{d}\boldsymbol{\xi}_{\mathrm{s}} \mathrm{d}\boldsymbol{\xi}_{\mathrm{s}}' \\
&\approx \underbrace{\sum_{j=1}^{M} \mathbf{w}_{i,j}}_{\text{Eq. (19)}} \underbrace{\sum_{t=1}^{M} \frac{\exp\left((\mathbf{W}_{\mathbf{q}} \boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},j}))(\mathbf{W}_{\mathbf{k}} \boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},t}))^{T}/\tau\right)}{\sum_{p=1}^{M} \exp\left((\mathbf{W}_{\mathbf{q}} \boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},j}))(\mathbf{W}_{\mathbf{k}} \boldsymbol{u}_{\mathrm{s}}(\boldsymbol{\xi}_{\mathrm{s},p}))^{T}/\tau\right)} \mathbf{W}_{\mathbf{v}} \left(\underbrace{\frac{\sum_{p=1}^{N} \mathbf{w}_{p,t} \boldsymbol{u}(\mathbf{g}_{p})}{\sum_{p=1}^{N} \mathbf{w}_{p,t}}}_{\text{Eq. (17)}} \right)}_{\text{Eq. (18)}} \\
&= \sum_{j=1}^{M} \mathbf{w}_{i,j} \sum_{t=1}^{M} \frac{\exp\left(\mathbf{q}_{j} \mathbf{k}_{t}^{T}/\tau\right)}{\sum_{p=1}^{M} \exp\left(\mathbf{q}_{j} \mathbf{k}_{p}^{T}/\tau\right)} \mathbf{v}_{t},
\end{aligned}
\tag{27}
$$

□

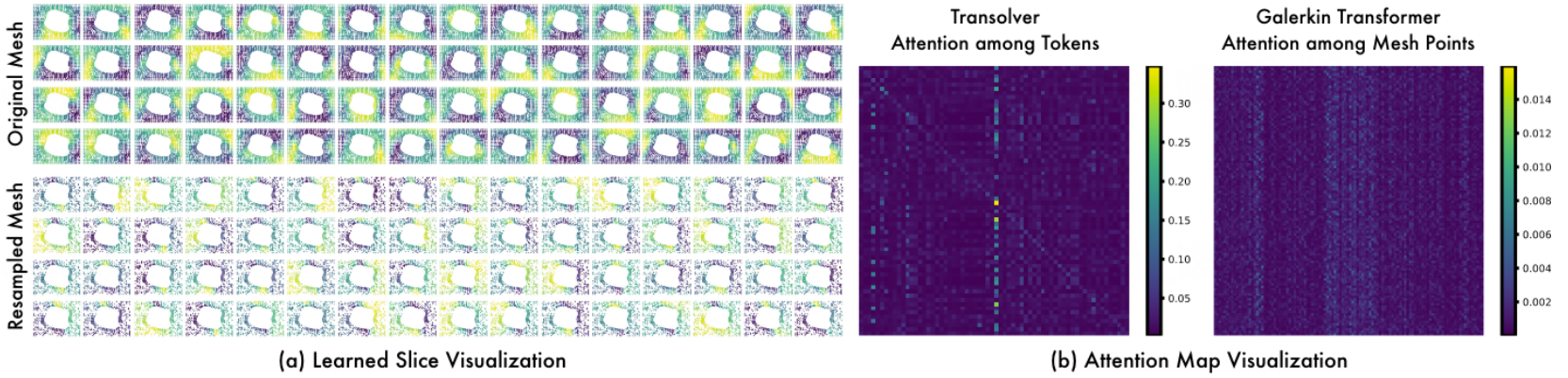


Figure 6. Physics-Attention visualization on Elasticity: (a) slice weights in the last layer of Transolver for both original and resampled meshes, (b) attention maps of the last layer in Transolver and Galerkin Transformer (Cao, 2021). See Appendix D.1 for more visualizations.

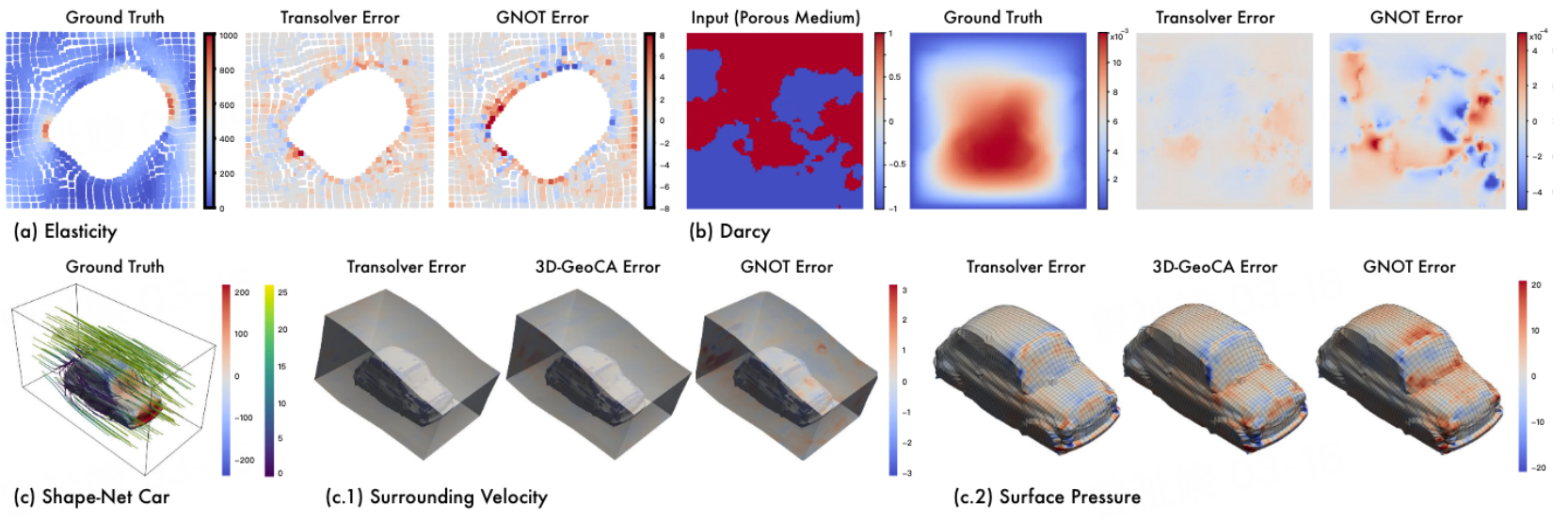


Figure 7. Case study on error maps of different models. Notably, Shape-Net Car requires to predict the surrounding velocity and surface pressure simultaneously. For clearness, we plot the error on volume and surface separately. See Appendix D.2 for more showcases.

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 (Anonymous, 2023a)	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**

model (1st The second best error) on each benchmark. / means that the baseline cannot apply to this benchmark.

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

9 Multiple Physics Pretraining

Problem: Let S be an arbitrary physics-driven spatiotemporal dynamical systems, either described by a parameterized family of PDEs with fixed parameters.

System: To simplify notation, we discuss systems with a single state variable in one spatial dimension. A continuous state variable for system S is represented as $u^S(x,t)$: $[0, L_S] \times [0, \infty) \to \mathbb{R}$. We discretize the system uniformly in space and time at resolutions N_S, T_S respectively.

Data: A snapshot $u_t^S \in \mathbb{R}^{N_S}$ represents the value of state variable u^S at all N_S spatial discretization points at time t.

Task: learn a single model $\mathcal{M}$ that can take a uniformly spaced sequence of T_S snapshots $\boldsymbol{u}_t^S = [u_{t-T_s\Delta t_S}^S, \ldots, u_t^S]$ from system S sampled from some distribution over systems and predict $\mathcal{M}(\boldsymbol{u}_t^S)$ such that $\mathcal{M}(\boldsymbol{u}_t^S) \approx u_{t+\Delta t_S}^S$.

9.1 Observation

1. Learning partially overlapping physics is beneficial for transfer learning
2. Single models can simultaneously learn many types of physics

Let $\psi(x,t)$ be a scalar defined on a periodic spatial domain, v a constant one-dimensional velocity coefficient and δ a constant diffusion coefficient, then:

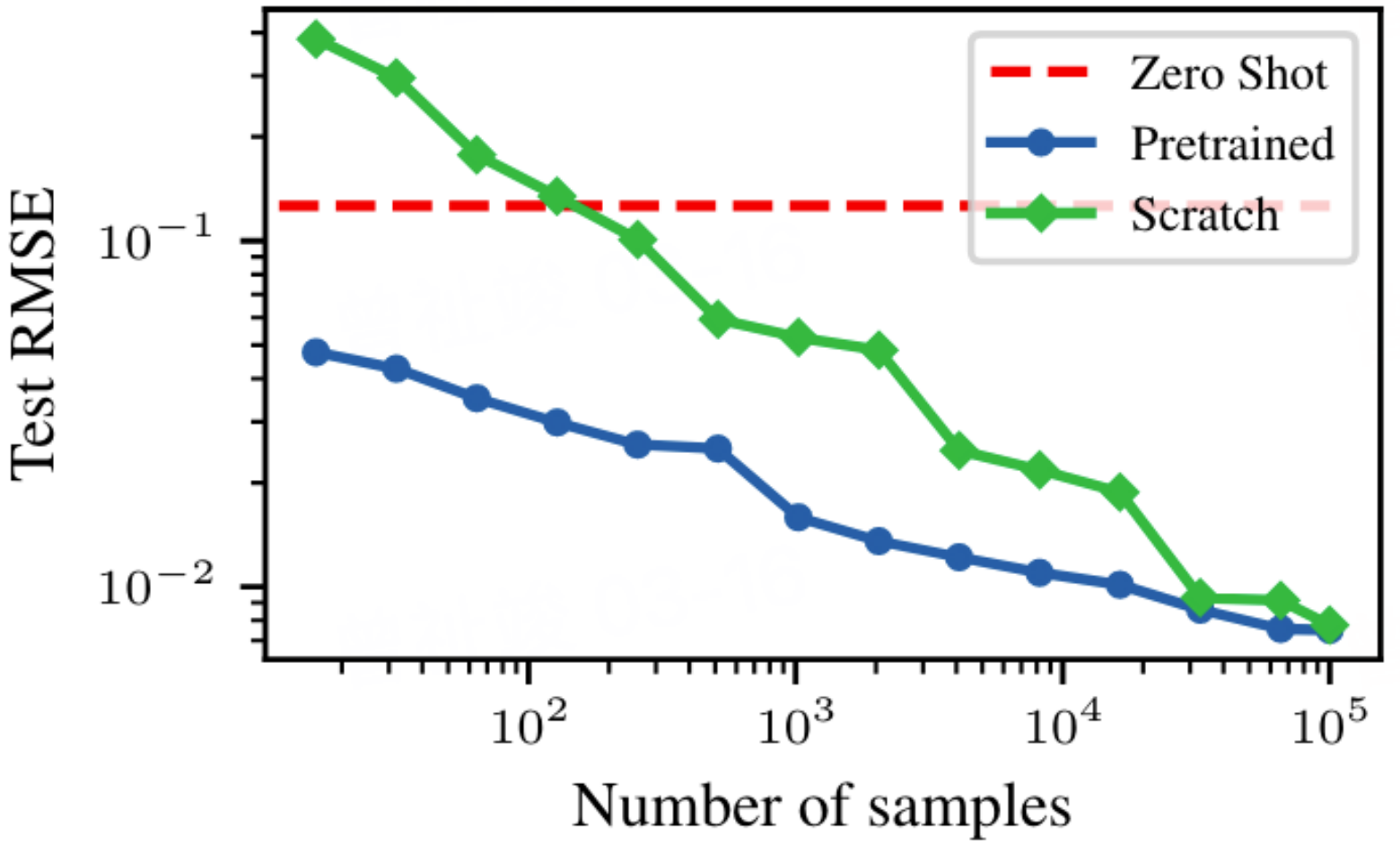
$$\textbf{Advection:} \quad \frac{\partial \psi}{\partial t} + \nabla \cdot (v\,\psi) = 0 \tag{28}$$

$$\textbf{Diffusion:} \quad \frac{\partial \psi}{\partial t} + \nabla \cdot (-\delta\,\nabla\,\psi) = 0 \tag{29}$$

$$\textbf{Advection-Diffusion:} \quad \frac{\partial \psi}{\partial t} + \nabla \cdot (v\,\psi - \delta\,\nabla\,\psi) = 0. \tag{30}$$

Experiment:

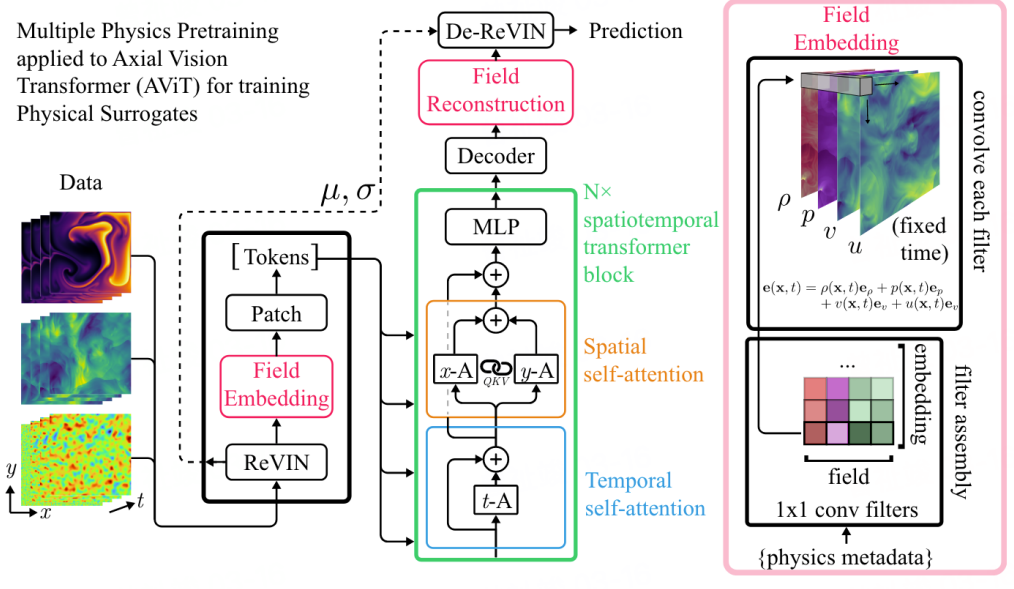
We pretrain a spatiotemporal transformer model on a large amount of trajectories (100, 000 each) with uniformly sampled coefficients ($v \in [-3, 3], \delta \in [10^{-3}, 1.]$) generated from the advection and diffusion equations while finetuning on restricted samples from advection-diffusion simulations.

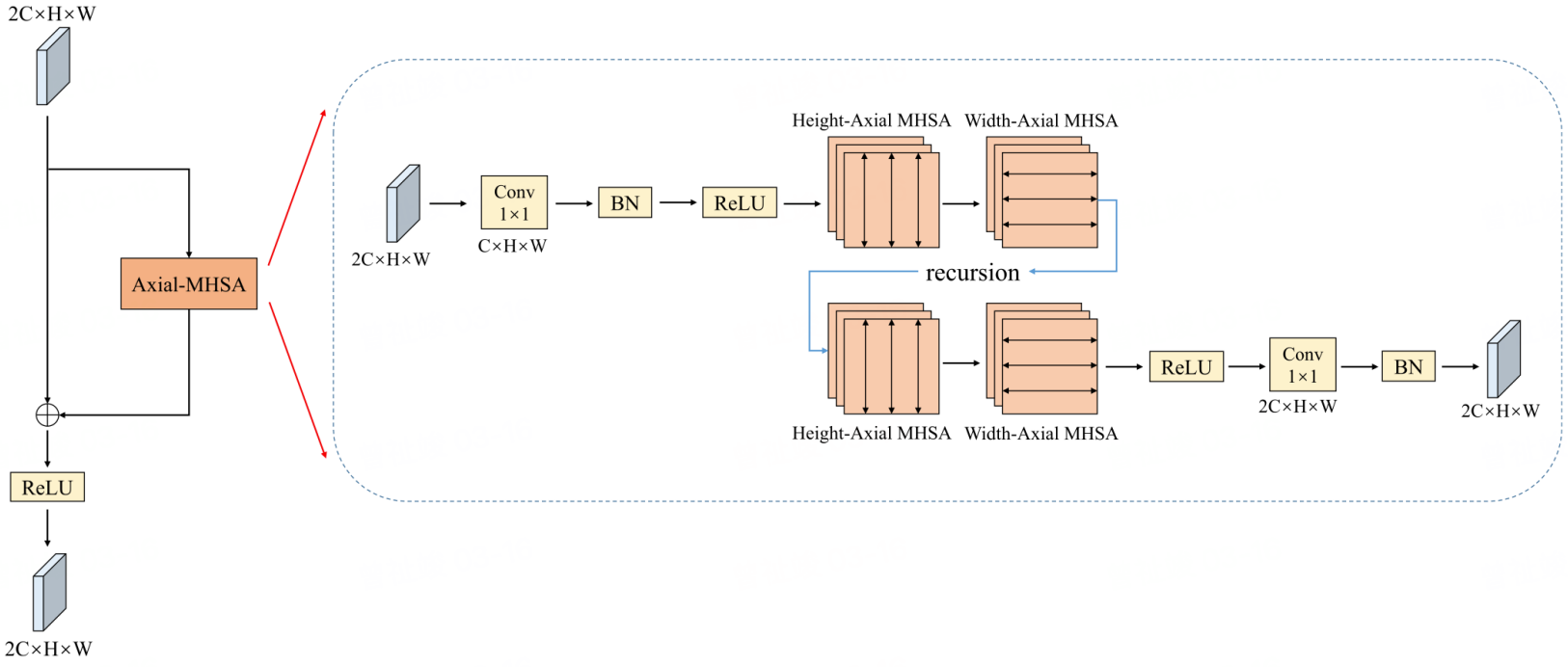

Zero Shot
Pretrained
Scratch
Test RMSE
10^{-1}
10^{-2}
10^{2}
10^{3}
10^{4}
10^{5}
Number of samples

9.2 Axial Attention

For a (2+1)-dimensional system with $T \times H \times W$ tokens, conventional dense attention attends over all tokens simultaneously and has cost $O((HWT)^2)$. Axial attention instead performs a series of attention operations over each axis in turn, limiting the cost to $O(H^2 + W^2 + T^2)$.

While we perform attention on each axis independently, spatial attention utilizes one set of linear projections for both the height (y) and width (x) axes.




2C×H×W
Axial-MHSA
ReLU
2C×H×W
2C×H×W
Conv
1×1
C×H×W
BN
ReLU
Height-Axial MHSA
Width-Axial MHSA
recursion
Height-Axial MHSA
Width-Axial MHSA
ReLU
Conv
1×1
2C×H×W
BN
2C×H×W

9.3 Field Embedding and Normalization

Normalization: To unify the magnitudes, we utilize reversible instance normalization . We compute the mean and standard deviation of each channel over the space- time dimensions and use them to normalize the input fields.

Embedding: the data is projected into a shared embedding space. This is the only component with weights that are unique to each source system. Given a system S with state variables $u(x,t), v(x,t), p(x,t) \in \mathbb{R}$, we project each sample point or "pixel" into a space of dimension D^{emb}:

$$\boldsymbol{e}(x,t) = u(x,t)\boldsymbol{e}_u + v(x,t)\boldsymbol{e}_v + p(x,t)\boldsymbol{e}_p \tag{31}$$

where $\boldsymbol{e}$ are embedding vectors in $\mathbb{R}^{D^{\text{emb}}}$.

Prediction: Prediction are reconstructed by sequence of transposed conv blocks and inner-product projection

$$u(x,t+\Delta t) = <\boldsymbol{e}(x,t+\Delta t), \boldsymbol{r}_u>$$

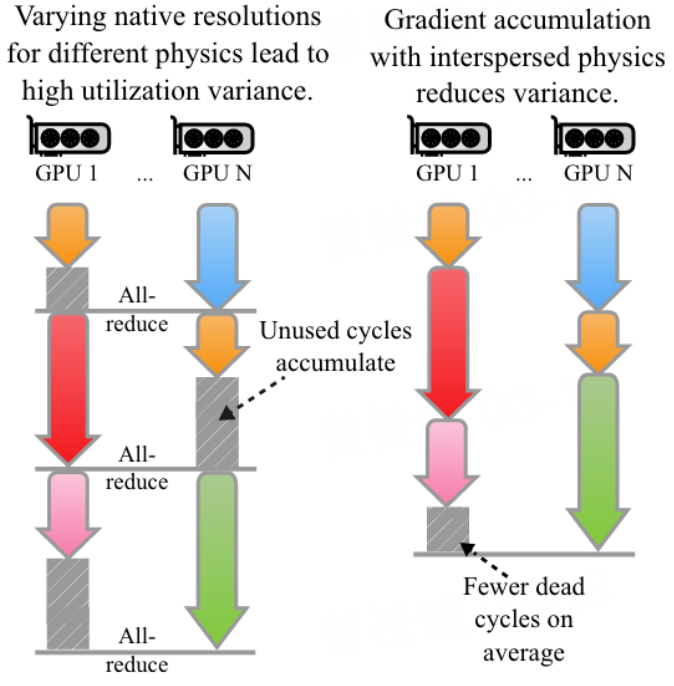
Position Biases: Using RoPE to embedding position inform. For periodic B.C we change the calculation of distance via periodic property.

9.4 Blancing Objectives

Our pretraining procedure operates on multiple levels of sampling. The task distribution varies in system S, spatial resolution N_S, and time resolution T_S.

Gradient accumulation utilizes multiple backward passes per synchronization step.

We therefore sample a single system S uniformly from $\mathcal{S}$ for each *micro-batch*. With m micro-batches per synchronization step, we reduce the work-per-GPU variance $\sigma^2_{\mathcal{B}}$ to $\frac{1}{m}\sigma^2_{\mathcal{B}}$, significantly reducing the average lost cycles due to work discrepancies.



9.5 Experiment

Can large transformer models learn the dynamics of multiple physical systems simultaneously?

Table 1: NRMSE comparison between MPP-pretrained models and dedicated baselines. MPP-pretrained models learn multiple physical systems at least as well as standard baselines. Top performing within size range and overall are bolded. Dashes indicate precision not available. † While the PINN is much smaller, these models are fit per-example.

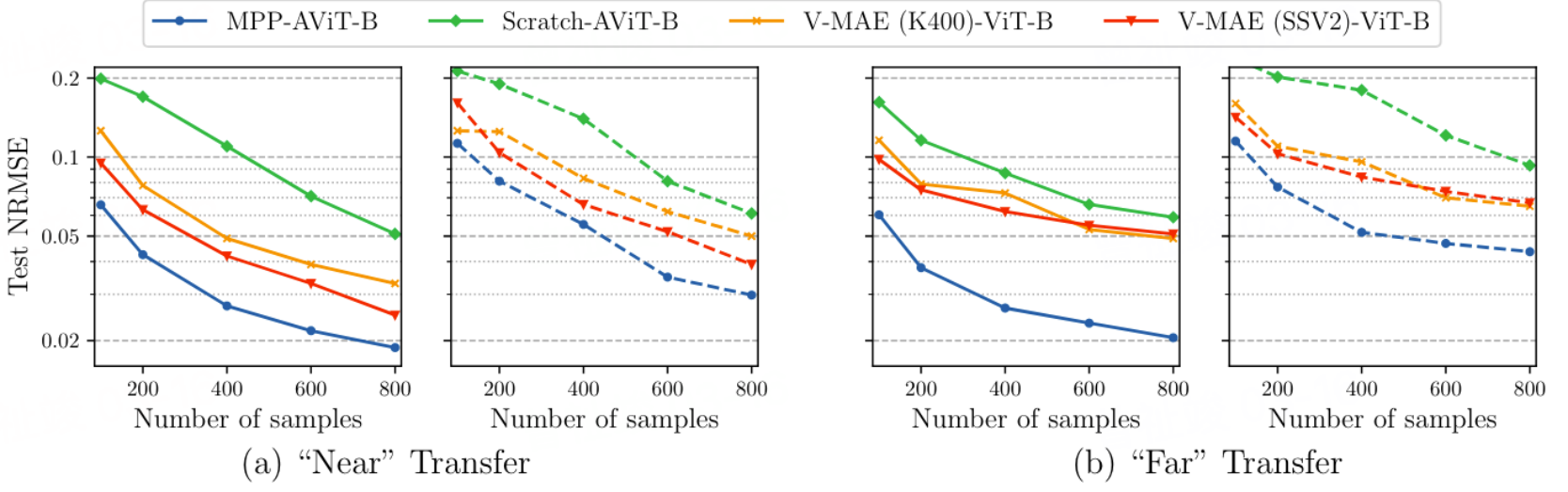
Model	#Param	SWE	DiffRe2D	CNS M1.0	CNS M0.1
MPP-AViT-Ti	7.6M	0.0066	**0.0168**	**0.0442**	**0.0312**
UNet	7.7M	0.083-	0.84–	0.4725	1.6650
FNO	466K	**0.0044**	0.12–	0.1685	0.2425
PINN	8.5K†	0.017-	1.6—	—	—
ORCA-SWIN-B	88M	0.0060	0.82–	—	—
MPP-AViT-B	116M	**0.0024**	**0.0106**	**0.0281**	**0.0172**
MPP-AViT-S	29M	0.0039	0.0112	0.0319	0.0213
MPP-AViT-L	409M	**0.0022**	**0.0098**	**0.0208**	**0.0147**

The improvement over the baselines is nearly an order of magnitude in NRMSE and the performance improves with scale.

Does MPP provide a finetuning advantage over existing spatiotemporal foundation models for new autoregressive prediction tasks?

We remove all compressible fluid data from the training corpus and pretrain on the three remaining spatiotemporal systems. We evaluate transfer to two specific compressible Navier-Stokes datasets:

- *"Near"*: $M = 0.1$, viscosity$=10^{-2}$, Random Periodic Initial Conditions
- *"Far"*: $M = 1.0$, viscosity$=10^{-8}$, Turbulent Initial Conditions



(a) "Near" Transfer

(b) "Far" Transfer

9.6 Broader Usage Of Pretrained Representation

Forcing Identification for Incompressible Navier-Stokes

We attempt to identify the constant forcing term used in the incompressible Navier-Stokes simulation from an input trajectory

Buoyancy for Incompressible Navier-Stokes

PDEArena includes an incompressible Navier-Stokes simulation with variable buoyancy. Since this set was not used during training, we take 1,000 randomly sampled trajectories for train, 100 for validation, and a further 1,000 for testing.

Table 3: RMSE for inverse problem tasks. Error from constant prediction included for context.

TRAINING	FORCING	BUOYANCY
MPP	$0.20^{\pm.008}$	$0.78^{\pm.006}$
SCRATCH	$0.43^{\pm.012}$	$0.77^{\pm.005}$
BEST CONSTANT	$1.00^{\pm.000}$	$0.77^{\pm.000}$

10 Flexible ML Models for Scientific Computing at Scale

10.1 Background

$\mathcal{E}$ denotes set of parametrized PDEs describing a spatialtemporal dynamical system $\mathcal{S}$. We represent the snapshots of the observed physical field within the system by $\mathcal{P}$. A continuous variable for $\mathcal{S}$ is defined as $u^{\mathcal{S}}(x,t)\colon \Omega \times [0,\infty] \to \mathbb{R}$. We discretized the system uniformly in space and time with resolution N_S and T_S then a snapshot $u_t^{\mathcal{S}} \in \mathbb{R}^{N_S}$ represent the value of state at time t.

1. Forward Problem: given equation $\mathcal{E}$, to predict the state of dynamical system $\mathcal{S}$ at any coordinate (x,t).
2. Inverse Problem: Given a set of PDEs $\tilde{\mathcal{E}}$, or in some case no explicit PDE information, along with observation comprising T_S snapshot $U_t^S = [u_1^S, \ldots, u_t^S]$, to infer the missing component(BC,IC,form of PDE) and reconstruct $\tilde{\mathcal{E}}$.

10.2 Method

10.2.1 Flexible Multiple Physics Pretraining

The framework is composed of a tokenization and encoding scheme, a state representation strategy for multichannel PDEs, an auto-regressive training mechanism, and a specialized optimization criterion.

Tokenization and Axis Encoding: Let $\mathcal{F}$ denotes a physical field, which is a function of space $\boldsymbol{x}$ and time t with channel C, an axis encoder $E: \mathbb{R}^M \rightarrow \mathbb{R}^H$ transform the physical field with dimension M to hiddien field with dimension H.

$$\boldsymbol{h}_{\boldsymbol{x},t} = E(\mathcal{F}(\boldsymbol{x}, t))$$

The hidden vectors $\boldsymbol{h}_{\boldsymbol{x},t}$ are flattened to form the input embeddings.

Channel-wise State Representation: For a PDE with multiple channels, we represent the state at t as a sequence grouped tokens $\boldsymbol{T}_t$ which includes C tokens

$$\mathbf{T}_t = [\mathbf{h}_{1,t}; \mathbf{h}_{2,t}; \ldots; \mathbf{h}_{C,t}]$$

This allows for the expression of interchannel and intra- channel relationships and dependencies on previous timesteps.

Auto-regressive Training with Shift-Righ: For a sequence of states $\boldsymbol{T}_{1:T}$ the model predicts the next state token $\hat{\boldsymbol{T}}_{t+1}$

$$\hat{\boldsymbol{T}}_{t+1} = G(\boldsymbol{T}_{1:T})$$

Optimization via Batch-wise nRMSE:

Let $\hat{\mathcal{F}}$ denote the predicted physical field, the NRMSE of batch B is

$$\mathrm{nRMSE}(B) = \frac{1}{|B|} \sqrt{\sum_{(\mathbf{x},t)\in B} \left(\frac{\mathcal{F}(\mathbf{x},t) - \hat{\mathcal{F}}(\mathbf{x},t)}{\sigma_{\mathcal{F}}} \right)^2}$$

where $\sigma_{\mathcal{F}}$ is the std of this batch.

Decoder with Axis Numerical Embedding: We utilize the inverse of the axis encoding function $E^{-1}: \mathbb{R}^H \to \mathbb{R}^D$ to reconstruct the physical field from hidden representation.

$$\hat{\mathcal{F}}(\boldsymbol{x}, t) = E(\hat{\boldsymbol{h}}_{\boldsymbol{x},t})$$

10.3 Physics-informed Reinforcement Learning

Let $\mathcal{D} = \{(\boldsymbol{x}_i, t_i)\}_{\{i=1\}}^{N}$ be the dataset containing pairs of physical field data and their corresponding weakly annotated PDE captions, here $\boldsymbol{x}_i$ represents a frame of physical field data and $\boldsymbol{t}_i$ is the textual equation, boundary conditions, and other related annotations in natural language.

Physics-informed Scorer Model S: A clip-style model S takes a pair of $\boldsymbol{x}, \boldsymbol{t}$ and output a similarity score $s \in \mathbb{R}$ indicating the match between the data and the textual information $S(\boldsymbol{x}, \boldsymbol{t}; \theta_S) \rightarrow s$

Pretrained Model G: Let G be a model that takes a sequence of recent time steps of physical fields $\boldsymbol{X}_{(0:t)} = \boldsymbol{x}_0, \boldsymbol{x}_{\delta_t}, \ldots, \boldsymbol{x}_t$ as input and predict physical field $\hat{\boldsymbol{x}}_{t+1}$

$$G(\boldsymbol{X}_{(0:t)}; \theta_G) \rightarrow \hat{\boldsymbol{x}}_{t+1}$$

Fine-Tuning with Physics-informed Rewards: During fine-tuning, input sequence $\boldsymbol{X}_{(0:t)}$ and its caption $\boldsymbol{t}$, the model G generates $\hat{\boldsymbol{x}}_{t+1}$, the scorer model S evalutate the reward $r = S(\hat{\boldsymbol{x}}_{t+1}, \boldsymbol{t}; \theta_S)$, the reward signal is used to update θ_G

$$\theta_G^* = \arg\max_{\theta_G} \mathbb{E}(\mathbf{X}_{(0:t)}), \mathbf{t} \sim \mathcal{D}[r]$$

10.4 Implementation

1. For generator, we use LLaMA model. For the 2D and 3D tasks, we apply a parallel convolutional encoder with the axis numerical encoder for better high- frequency pattern capturability.For the decoder, we use the sum of logits from both transposed convolutional decoder and the weights-bined inverse axis numerical decoder.

2. Equation Rewriting: We apply mathematical identities to modify the equation, ensuring the core properties remain intact.

3. Linear Combination: For systems of equations, we derive new variants through linear combinations, enriching the dataset without altering the system's nature.

4. Symbol Substitution: We systematically swap variables with alternative symbols, such as replacing x with ξ , to maintain consistency and avoid ambiguity.

5. Physical Checking: A panel of GPT- 4- based experts evaluates the augmented equations, filtering out those that do not align with physical principles.

A.2.1. BURGERS 1D

- Original form:

$$\partial_t u(t,x) + \partial_x(u^2(t,x)/2) = \nu/\pi \partial_{xx} u(t,x), \quad x \in (0,1), t \in (0,2],$$
$$u(0,x) = u_0(x),\ x \in (0,1),$$

- After augmented:

$$0.77 \int \left(\frac{\partial}{\partial t} v(t,x) + \frac{\partial}{\partial x} \frac{v^2(t,x)}{2} \right) dt = \frac{0.77\nu \int \frac{\partial^2}{\partial x^2} v(t,x)\, dt}{\pi}$$

$$0.73tv(0,x) = 0.73tv_0(x)$$

- Explanation: We replace u with v and ∂_t with $\frac{\partial}{\partial t}$. We integrate and multiply some factors on both sides of the equation at the same time.

A.2.3. CFD 1D

- Original form:

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0,$$
$$\rho(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}) = -\nabla p + \eta \triangle \mathbf{v} + (\zeta + \eta/3)\nabla(\nabla \cdot \mathbf{v}),$$
$$\partial_t \left[\epsilon + \frac{\rho v^2}{2} \right] + \nabla \cdot \left[\left(\epsilon + p + \frac{\rho v^2}{2} \right) \mathbf{v} - \mathbf{v} \cdot \sigma' \right] = 0,$$

- After augmented:

$$\varrho(t,x) \frac{\partial}{\partial x} \mathbf{w}(t,x) + \frac{\partial}{\partial t} \varrho(t,x) = 0$$

$$0.61 \left(\mathbf{w}(t,x) \frac{\partial}{\partial x} \mathbf{w}(t,x) + \frac{\partial}{\partial t} \mathbf{w}(t,x) \right) \varrho(t,x) = 0.61\eta \frac{\partial^2}{\partial x^2} \mathbf{w}(t,x) + 0.61 \left(\chi + \frac{\eta}{3} \right) \frac{\partial^2}{\partial x^2} \mathbf{w}(t,x) - 0.61 \frac{\partial}{\partial x} p(t,x)$$

$$\int \left(\frac{\partial}{\partial t} \left(\varepsilon + \frac{\mathbf{w}^2(t,x)\varrho(t,x)}{2} \right) + \frac{\partial}{\partial x} \left(-\tau \mathbf{w}(t,x) + \left(\varepsilon + \frac{\mathbf{w}^2(t,x)\varrho(t,x)}{2} + p(t,x) \right) \mathbf{w}(t,x) \right) \right) dt = 0$$

- Explanation: We replaced many symbols, such as replacing ∇ with ∂_t and $\triangle$ with $\frac{\partial^2}{\partial x^2}$,. We integrate and multiply some factors on both sides of the equation at the same time. We also swapped the order of some items, such as $\zeta + \eta/3$.

The text encoder leverages the pre-trained albert-math model, which is adept at processing LaTeX-encoded PDE captions due to its extensive training on a large corpus of LaTeX data.

For the physics encoder, we implement a streamlined one-channel Vision Transformer (ViT) model tailored to both 1D and 2D PDEs, and we adopt a large-batch contrastive learning approach, akin to the CLIP framework.

PDEs		FNO	PINNs	U-Net	MPP-L	OmniArch	+ PIRL	Lead %
1D	**CFD**	1.4100	-	2.6700	-	0.0981	**0.0392**	97.2 ↑
	ReacDiff	0.0005	0.2140	0.0026	-	0.0004	**0.0002**	64.0 ↑
	Advection	0.0091	0.8130	0.7760	-	0.0081	**0.0045**	49.9 ↑
	Burgers	0.0174	0.9450	0.3200	-	0.0067	**0.0032**	81.9 ↑
	Diff_Sorp	0.0017	0.2200	0.1500	-	0.0019	**0.0006**	62.2 ↑
2D	**CFD**	0.2060	-	1.0700	0.0178	0.0994	**0.0153**	14.0 ↑
	SWE	0.0044	0.0170	0.0830	0.0022	0.0031	**0.0015**	3.32 ↑
	ReacDiff	0.1200	1.6000	0.8400	0.0098	0.0993	**0.0084**	14.0 ↑
	NS-Incom	0.2574	-	1.1200	-	0.1494	**0.0827**	67.9 ↑
3D	**CFD**	0.3050	-	0.6150	-	0.6600	0.5072	66.3 ↓

Table 3: The Peroformace on Zero-Shot PDEs.

Methods	Shock	KH	OTVortex
FNO	0.7484	1.0891	0.5946
U-Net	1.6667	0.1677	0.4217
MPP	0.3243	1.3261	0.3025
OmniArch	**0.2126**	**0.5432**	**0.1718**

- **OTVortex:** The Orszag-Tang Vortex system is a compressible flow problem that generates highly complex vortex structures through the careful selection of initial conditions. The dataset includes one exsample, which is a 1024×1024 resolution physical field evolved over 101 time steps with a time interval of 0.01.

- **2D Shock:** Shock waves are characterized by abrupt changes in flow properties resulting from sudden discontinuities in fluid flow, such as rapid changes in pressure, temperature, and density. The dataset includes one exsample, which is also a 1024×1024 resolution physical field evolved over 101 time steps with a time interval of 0.01.

- **2D KH:** The Kelvin-Helmholtz instability is a fluid instability that occurs at the interface between two fluid layers with different velocities or densities. This dataset consists of seven exsamples generated based on different parameters M, dk, and Re. Each is a 1024×1024 resolution physical field evolved over 51 time steps with a time interval of 0.1. We conducted experiments on all samples and averaged the results.

11 CoDA-NO

11.1 Preliminery

For any input function $a: \mathcal{D} \to \mathbb{R}^{d_{in}}$, we will denote the d_{in} dimensional output shape as codomain. We consider each of the components of the codomain as different physical variables, which are real-valued functions over the input domain $\mathcal{D}$,i.e., $a = [a^1, \ldots, a^{d_{in}}]$ with $a^i: \mathcal{D} \to \mathbb{R}$. The same applies to the output function $u: \mathcal{D} \to \mathbb{R}^{d_{out}}$.

A Neural Operator seeks to approximate an operator G that maps an input function $a \in \mathcal{A}$ to its corresponding output function $u \in \mathcal{U}$ by building a parametric map $\mathcal{G}_\phi: \mathcal{A} \to \mathcal{U}$. The typical architecture of a Neural Operator can be described as

$$\mathcal{G}_\phi = \mathcal{P} \circ \mathcal{I}_L \circ \ldots \mathcal{I}_1 \circ \mathcal{L}. \tag{32}$$

Here, $\mathcal{L}: a \to w_0$ and $\mathcal{P}: w_L \to u$ are lifting and pointwise projection operators, respectively. The action of any pointwise operator $\mathcal{H}: \{f: \mathcal{D} \to \mathbb{R}^{d_f}\} \to \{g: \mathcal{D} \to \mathbb{R}^{d_g}\}$ can be defined as

$$\mathcal{H}[f][x] = h_\phi(f(x)), \tag{33}$$

where $h_\phi: \mathbb{R}^{d_f} \to \mathbb{R}^{d_g}$ is any function with parameters ϕ.

The integral operator $\mathcal{I}_l: w_{l-1} \to w_l$ performs a kernel integration over the input function w_{l-1} as

$$\mathcal{I}_l[w_{l-1}](x) = \int_{\mathcal{D}_{l-1}} k_l(x, y)\, w_{l-1}(y)\, \mathrm{d}y. \tag{34}$$

Here, $\mathcal{D}_{l-1}$ is the domain of the function w_{l-1}. In the case of Fourier Neural operators (FNO) , a convolution kernel, $k_l(x, y) = k_l(x - y)$ was used. By the convolution theorem, this enables the representation of an integral operator as a pointwise multiplication of the Fourier coefficients as follows:

$$w_l = \mathbb{F}^{-1}(\mathbb{F}(k_l) \odot \mathbb{F}(w_{l-1})). \tag{35}$$

For Graph neural operator (GNO) , a small neighborhood $B_r(x) \cap \mathcal{D}_{l-1}$ around the point x is considered instead of integrating over the whole domain $\mathcal{D}_{l-1}$, such that the equation changes to

$$w_l(x) = \int_{B_r(x) \cap D_{l-1}} k_l(x, y)\, w_{l-1}(y)\, \mathrm{d}y. \tag{36}$$

Given a set of evaluations of the function w_{l-1} on points $\{y_i\}_{i=1}^n \subset \mathcal{D}_{l-1}$, the kernel integral can be approximated by

$$w_l(x) \approx \sum_{y_i \in B_r(x)} k_l(x, y_i)\, w_{l-1}(y_i)\, q_i, \tag{37}$$

where $q_i \in \mathbb{R}$ are suitable quadrature weights. The discretized kernel integral can be viewed as a message passing on graphs, where the neighborhood of each point x consists of all points within radius r.

12 Method

12.1 Problem statement

Let's consider two input functions $a: \mathcal{D} \to \mathbb{R}^{d_{in}}$ and $\tilde{a}: \mathcal{D} \to \mathbb{R}^{\tilde{d}_{in}}$ of two different PDE solution operators with corresponding output functions $u: \mathcal{D} \to \mathbb{R}^{d_{out}}$ and $\tilde{u}: \mathcal{D} \to \mathbb{R}^{\tilde{d}_{out}}$. In general, the functions a and $\tilde{a}$ represent d_{in} and $\tilde{d}_{in}$ physical variables over the domain $\mathcal{D}$ with $d_{in} \neq \tilde{d}_{in}$. We aim to design neural operator architectures $\mathcal{G}$ that can both be applied to a as well as $\tilde{a}$ despite the different codomains of the input as well as output functions.

In particular, when the PDE systems have overlapping physical variables $\{a^i\}_{i=1}^{d_{in}} \cap \{\tilde{a}^i\}_{i=1}^{\tilde{d}_{in}} \neq \emptyset$, this naturally allows to transfer learned knowledge from one system to the other.

12.2 Permutation Equivariant Neural Operator

As we consider the vector-valued input function a as a set of d_{in} functions $\{a^1, a^2, \ldots, a^{d_{in}}\}$ that represent different physical variables of the PDE.

For an efficient implementation, we mimic transformer architectures and share weights across different variables. We achieve this by defining *permutation equivariant integral operator* $\mathcal{I}_{per}$ as

$$\mathcal{I}_{per}[w] = [\mathcal{I}[w_e^1], \ldots, \mathcal{I}[w_e^{d_{\text{in}}}]], \tag{38}$$

where I is a regular integral operatorand w_e^i is the codomain group of the input variable i. Following the same mechanism, we can also define permutation equivariant pointwise operator $\mathcal{H}_{per}$ with a shared pointwise operator $\mathcal{H}$. We will use FNO$_{per}$ and GNO$_{per}$ to denote permutation equivariant operators using a shared GNO and FNO, respectively.

12.3 CoDA-NO Layer

Given a function $w: \mathcal{D} \to \mathbb{R}^d$, we partition the function into a set of so-called *token functions* $w^i: \mathcal{D} \to \mathbb{R}^{d'}$ for $i \in \{1, \dots L\}$ along the codomain, such that $w = [w^1, \dots w^T]$. That is, w represents the codomain-wise concatenation of the token functions w_i and $d' = \frac{d}{L}$. If no other value is specified, we assume that $d' = 1$.

Single-head CoDA-NO layer: We extend the key, query, and value *matrices* of the standard attentio to *operators* mapping token functions $w^i: \mathcal{D} \to \mathbb{R}^{d'}$ to key, query, and value functions. We define the key, query, and value operators as

$$\mathcal{K}: w^i \to \{k^i: \mathcal{D} \to \mathbb{R}^{d_k}\}, \tag{39}$$
$$\mathcal{Q}: w^i \to \{q^i: \mathcal{D} \to \mathbb{R}^{d_q}\}, \tag{40}$$
$$\mathcal{V}: w^i \to \{v^i: \mathcal{D} \to \mathbb{R}^{d_v}\}. \tag{41}$$

Assuming $d_k = d_q$, we denote by $k^i = \mathcal{K}[w^i]$, $q^i = \mathcal{Q}[w^i]$, and $v^i = \mathcal{V}[w^i]$ the key, query, and value functions of the token functions, respectively.

Next, we calculate the output (token) functions $o^i \colon \mathcal{D} \to \mathbb{R}^{d_v}$ as

$$o^i = \texttt{Softmax}\left([\frac{\langle q^i, k^1 \rangle}{\tau} \dots \frac{\langle q^i, k^T \rangle}{\tau}] \right) [v^1, \dots, v^L]^\top \tag{42}$$

where τ is the *temperature* hyperparameter. Here, $\langle ., . \rangle$ denotes a suitable dot product in the function space. We take the $L^2(\mathcal{D}, \mathbb{R}^{d_k})$-dot product given by

$$\langle q^i, k^j \rangle = \int_{\mathcal{D}} \langle q^i(x), k^j(x) \rangle \, \mathrm{d}x, \tag{43}$$

where the integral can be discretized using quadrature rules, similar to the integral operator.

To implement multi-head attention, we apply the (single-head) attention mechanism described above separately for multiple heads $h \in \{1, \dots H\}$ using $\mathcal{K}^h$, $\mathcal{Q}^h$, and $\mathcal{V}^h$ to obtain $o^{i,h}$. We then concatenate these outputs $o^{i,h}$ along the codomain and get $c^i := [o^{i,1}, \dots o^{i,H}]$. Finally, we use an operator

$$\mathcal{M} \colon \{c^i \colon \mathcal{D} \to \mathbb{R}^{H \cdot d_v}\} \to \{o^i \colon \mathcal{D} \to \mathbb{R}^{d_v}\} \tag{44}$$

to get the output function o^i.

We concatenate o^i as $o = [o^1, o^2, \dots o^L]$ to obtain multi-head attention.

12.4 Function Space Normalization

Given a function w, let $w_e^j \colon \mathcal{D} \to \mathbb{R}^{d_l}$ be the codomain group. Then we calculate the mean $\mu \in \mathbb{R}^{d_l}$ and standard deviation $\sigma \in \mathbb{R}^{d_l}$ for each codomain group as

$$\mu^j = \int_{\mathcal{D}} w_e^j(x)\,\mathrm{d}x. \tag{45}$$

$$\sigma^j = \left(\int_{\mathcal{D}} (w_e^j(x) - \mu^j)^{\circ 2}\,\mathrm{d}x \right)^{\circ \frac{1}{2}}. \tag{46}$$

Here, $\circ i$ denotes the elementwise (Hadamard) i^{th}-power. The normalization operator can be written as

$$\texttt{Norm}[w_e^j](x) = (\boldsymbol{g} \oslash \sigma^j) \odot (w_e^j(x) - \mu^j) + \boldsymbol{b}, \tag{47}$$

where $\boldsymbol{g} \in \mathbb{R}^{d_l}$ and $\boldsymbol{b} \in \mathbb{R}^{d_l}$ are learnable bias and gain vectors and $\oslash$ and $\odot$ denote elementwise division and multiplication operation. This normalization can be seen as an extension of the Instance Normalizatio for function spaces.

12.5 Variable Specific Positional Encoding (VSPE)

We learn positional encoders $e^j: \mathcal{D} \to \mathbb{R}^{d_{en}}$ for each physical variable $j \in \{1, \ldots, d_{in}\}$, for the given vector-valued input function $a = [a^1, \ldots, a^{d_{in}}]$. We concatenate each positional encoding e^j with the respective variable $a^j: \mathcal{D} \to \mathbb{R}$ along to codomain to obtain extended input functions $a_e^j = [a^j, e^j]$. Next, we apply a shared pointwise lifting operator

$$\mathcal{P}: \{a_e^j: \mathcal{D} \to \mathbb{R}^{d_{en}+1}\} \to \{w_e^j: \mathcal{D} \to \mathbb{R}^d\}, \tag{48}$$

typically with $d > d_{en} + 1$. Finally, we concatenate w_e^j, $j \in \{1, \ldots d_{in}\}$, to get the lifted function

$$w = [w_e^1, \ldots, w_e^{d_{in}}]: \mathcal{D} \to \mathbb{R}^{d \cdot d_{in}}. \tag{49}$$

We refer to each w_e^j as a *codomain group*.

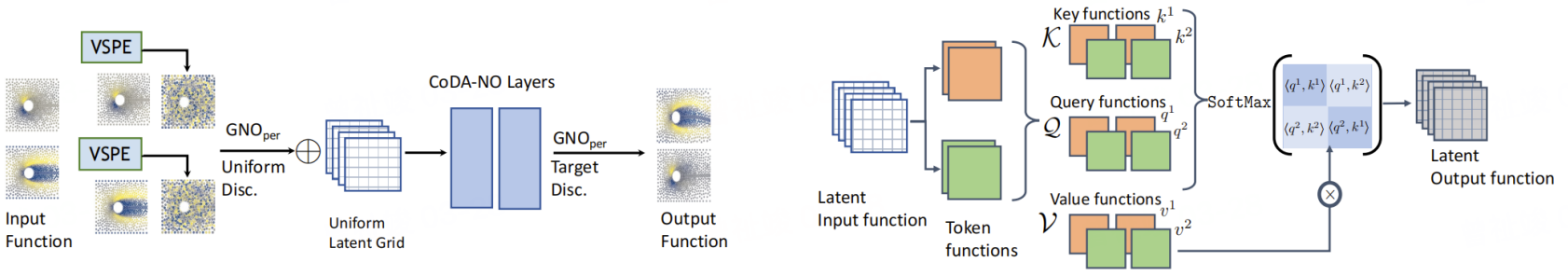


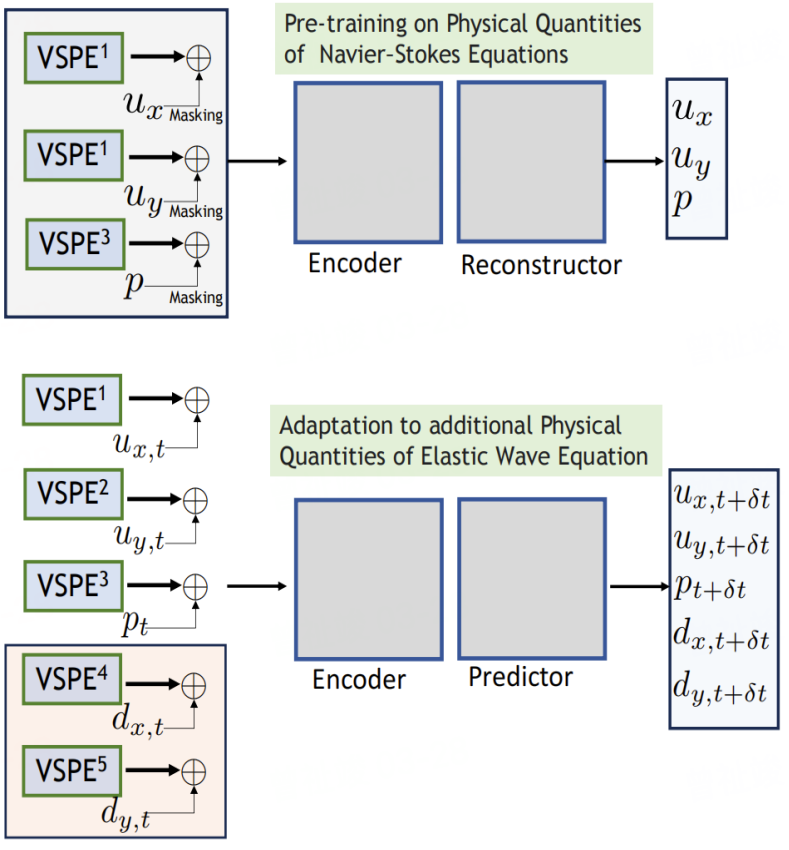
Figure 2. On the **left**, we illustrate the architecture of the Codomain Attention Neural Operator. Each physical variable (or co-domain) of the input function is concatenated with *variable specific positional encoding* (VSPE). Each variable, along with the VSPE, is passed through a GNO layer, which maps from the given non-uniform geometry to a latent regular grid. Then, the output on a uniform grid is passed through a series of CoDA-NO layers. Lastly, the output of the stacked CoDA-NO layers is mapped onto the domain of the output geometry for each query point using another GNO layer. On the **right**, we illustrate the mechanism of codomain attention. At each CoDA-NO layer, the input function is tokenized codomain-wise, and each token function is passed through the $\mathcal{K}$, $\mathcal{Q}$, and $\mathcal{V}$ operators to get key, query, and value functions $\{k^1, k^2\}$, $\{q^1, q^2\}$, and $\{v^1, v^2\}$ respectively. The output function is calculated via an extension of the self-attention mechanism to the function space.

13 Model Training

The model undergoes a two-stage training process: Self-supervised pretraining is followed by a supervised fine-tuning stage.

Pre-training. In the context of self-supervised pretraining, the objective is to train the model to reconstruct the original input function from its masked version. Within this phase, the model's encoding component is denoted as the *Encoder*, while the decoding component comprises the *Reconstructor*. The values of the input function at a specific percentage of mesh points are randomly masked to zero, and certain variables (of the co-domain) of the input function are entirely masked to zero. The model is then trained to reconstruct the original input from this masked version. We emphasize that the self-supervised learning phase is agnostic of the downstream supervised task and only requires snapshots of simulations of the physical systems.

Fine-tuning. In the supervised fine-tuning phase, the *Reconstructor* is omitted from the decoding module and replaced by a randomly initialized Predictor module. The parameters of the Encoder and VSPEs are copied from pre-trained weights. If the fine-tuning (target) PDE introduces variables that are not present in the pre-training PDE, we train additional variable encoders for these newly introduced variables . This ensures that the model adapts to the expanded set of variables needed for the fine-tuning task.


Pre-training on Physical Quantities of Navier-Stokes Equations
VSPE1
VSPE1
VSPE3
Masking
Masking
Masking
Encoder
Reconstructor
Adaptation to additional Physical Quantities of Elastic Wave Equation
VSPE1
VSPE2
VSPE3
VSPE4
VSPE5
Encoder
Predictor

14 Experiment

We consider the following problems: (a) a fluid dynamics problem, where a Newtonian, incompressible fluid impinges on a rigid object, and (b) a fluid- structure interaction problem between a Newtonian, incompressible fluid and an elastic, compressible solid object. The dynamics of the fluid are governed by the Navier-Stokes equations

$$\rho^f \frac{\partial u}{\partial t} + \rho^f \nabla \cdot (u \otimes u) = \nabla \cdot \boldsymbol{\sigma}^f, \; \nabla \cdot u = 0$$

where u and ρ^f denote the fluid velocity and density, respectively. And $\boldsymbol{\sigma}^f$ denotes the Cauchy stress tensor, given by

$$\boldsymbol{\sigma}^f = -p\,\mathbb{I} + \mu\,(\nabla u + \nabla u^T), \tag{50}$$

where $\mathbb{I}$ is the identity tensor, p the fluid pressure, and μ the fluid dynamic viscosity.

For fluid-structure interaction, the deformable solid is governed by the elastodynamics equations

$$\rho^s \frac{\partial^2 d}{\partial t^2} = \nabla . (J \boldsymbol{\sigma}^s (\mathbf{F}^{-1})^T) \qquad \text{in } \Omega_t^s \tag{51}$$

with $\mathbf{F} = \mathbb{I} + \nabla d$ and $J = \det(\mathbf{F})$. Here d, ρ^s, F, and $\boldsymbol{\sigma}^s$ denote the deformation field, the solid density, the deformation gradient tensor, and the Cauchy stress tensor.

Table 1. **Test errors (L_2 loss) for fluid dynamics (NS) and fluid-solid interaction (NS+EW) datasets** with viscosity $\mu = 5.0$ for different numbers of few-shot training samples.

Model	Pretrain Dataset	# Few Shot Training Samples 5		25		100	
		NS	NS+EW	NS	NS+EW	NS	NS+EW
GINO	-	0.200	0.122	0.047	0.053	0.022	0.043
DeepO	-	0.686	0.482	0.259	0.198	0.107	0.107
GNN	-	0.038	0.045	0.008	0.009	0.008	0.009
Ours	NS	0.025	0.071	0.007	0.008	**0.004**	0.005
	NS+EW	**0.024**	**0.040**	**0.006**	**0.005**	0.005	**0.003**

Table 2. **Ablation studies on fluid dynamics (NS) and fluid-solid interaction (NS+EW) datasets** with viscosity $\mu = 5.0$ for different numbers of few-shot training samples; Reporting L_2 loss.

Model	Pretrain Dataset	# Few Shot Training Samples 5		25		100	
		NS	NS+EW	NS	NS+EW	NS	NS+EW
ViT	-	0.271	0.211	0.061	0.113	0.017	0.020
U-Net	-	13.33	3.579	0.565	0.842	0.141	0.203
Ours	-	0.182	0.051	0.008	0.084	0.006	0.004
Ours	NS	0.025	0.071	0.007	0.008	**0.004**	0.005
	NS+EW	**0.024**	**0.040**	**0.006**	**0.005**	0.005	**0.003**

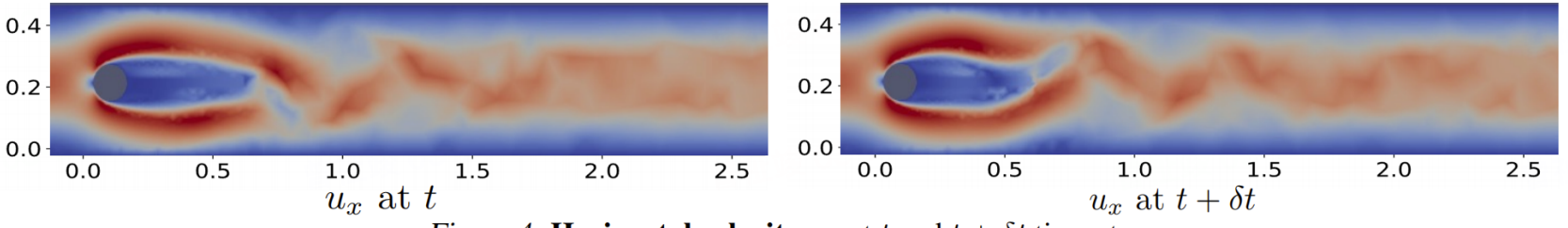


Figure 4. **Horizontal velocity** u_x at t and $t + \delta t$ time step.