Choose a Transformer: Fourier or Galerkin

Shuhao Cao

马珩元，诸格慧明，张崧

Contents

Contents

Transformer

A single attention head[1]

$$\text{Input } y \in \mathbb{R}^{n\times d}$$

$$\text{Parameters } W^Q, W^K, W^V \in \mathbb{R}^{d\times d}$$

$$Q = yW^Q, K = yW^K, V = yW^V \in \mathbb{R}^{n\times d}$$

$$\text{Attn}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V$$

n **denotes the length of the input sequence that can vary**, and d denotes the feature dimension of terms in the sequence.

[1] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems. 2017.

Achievements

Applications:

- NLP: BERT (Bidirectional Encoder Representations from Transformers)[2]
- CV:ViT (Vision Transformer)[3]
- Graph: GAT (Graph Attention Network)[4]
- ...

[2]Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[3]Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

[4]Veličković, Petar, et al. "Graph attention networks." arXiv preprint arXiv:1710.10903 (2017).

Weakness

Interpretability: why the Transformer performs well?
Doubt: Attention is not all you need![5]
Complexity: matrix multiplication, softmax operation, $O(n^2)$

[5] Dong, Yihe, Jean-Baptiste Cordonnier, and Andreas Loukas. "Attention is not all you need: Pure attention loses rank doubly exponentially with depth." arXiv preprint arXiv:2103.03404 (2021).

Galerkin Method

Supposed we want to solve a PDE on a finite domain Ω

$$\begin{aligned} -\Delta u(x) + u(x) &= f(x), x \in \Omega \\ u(x) &= 0, x \in \partial\Omega \\ u &\in \mathcal{Q} \end{aligned}$$

Consider the correponding variational problem

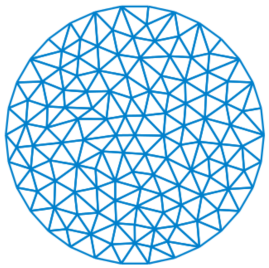
$$\begin{aligned} \int_\Omega (-\Delta u + u) v dx &= \int_\Omega f v dx \\ a(u, v) &=< f, v >, v \in \mathcal{Q} \end{aligned}$$

where $a(u, v) = \int_\Omega (-\Delta u + u) v dx$, and $< f, v >= \int_\Omega f v dx$.

Galerkin Method

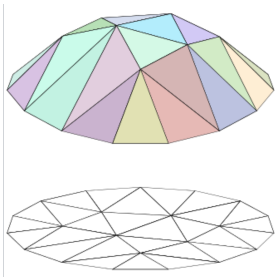
Discretize the domain Ω and approximate the functions in $\mathcal{Q}$ by some base functions.

$$\Omega$$



$$\mathcal{Q} \to \mathcal{Q}_h$$

$$u \to u_h$$



Galerkin Method

Choose a set of linear bases $\phi_i, i = 1, ..., n$ for $\mathcal{Q}_h$, then write the solution as $u = \sum_{i=1}^{n} u_i \phi_i$, where $u_i, i = 1, ..., n$ are the coefficients we need to determined.

Now, we only need to solve the linear system

$$\sum_{j=1}^{n} a(\phi_j, \phi_i) u_j =< f, \phi_i >, i = 1, ..., n$$

Céa Lemma

We have used u_h to approximate the solution of the following variational problem

$$a(u, v) =< f, v >, v \in \mathcal{Q},$$

the lemma says under some conditions of a and f, for a given size of grid $h = \frac{1}{n}$, we have

$$\|u - u_h\| \leq c(h) \min_{w \in \mathcal{Q}} \|u - w\| .$$

Noticed that $c(h)$ is dependent to h.

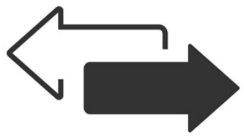
Thinking from a New Angle

Choose a Transformer: Fourier or Galerkin

Shuhao Cao
Department of Mathematics and Statistics
Washington University in St. Louis
s.cao@wustl.edu

Understand the Transformer under the perspective of numerical analysis (finite element method).

$$\text{softmax}(\frac{QK^T}{\sqrt{d}})V$$



$$\sum_{j=1}^{n} a(\phi_j, \phi_i) u_j =< f, \phi_i >$$

Contents

Operator learning involving parametric PDEs

Problem: find u in some functions space, such that $L_a(u) = f$, given the parameter a and f.
Equation solving: given $a = a_0$, solve the equation $L_{a_0}(u) = f$
Operator learning: Given pairs of data

$$\{(u^{(i)}, a^{(i)}) \mid L_{a^{(i)}}(u^{(i)}) = f, i = 1, ..., N\},$$

learn an operator T such that $T(a) = u$ for any a in the parameter space.

Operator Learning Involving Parametric PDEs

To handle the infinite dimension element, we approximate every function u by the vector u_h defined at a discrete grid of size $h \ll 1$. Then we define the model T_θ with learnable parameter θ. We train T_θ to approximate T **independent of** the mesh size h, the empirical loss function is

$$\mathcal{J}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \left[\left\| T_\theta(a^{(i)}) - u_h^{(i)} \right\|^2 + R(a^{(i)}, u_h^{(i)}, \theta) \right],$$

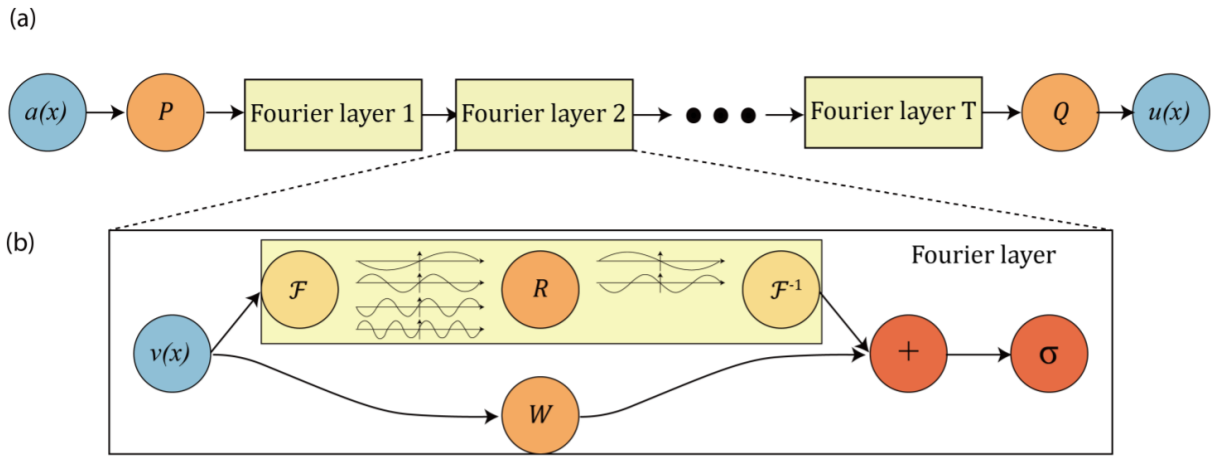
where $R(\cdot)$ is the regularization term.

Operator Learning Involving Parametric PDEs

An example: Fourier Neural Operator (FNO)[6]

$$(\mathcal{K}(\theta)v_t)(x) := \mathcal{F}^{-1}(R_\theta \cdot \mathcal{F}v_t))(x)$$
$$v_{t+1}(x) = \sigma(Wv_t(x) + (\mathcal{K}(\theta)v_t)(x)).$$

This operator encodes the input function on its frequency domain instead of time (or space) domain. By the decay of the Fourier series, only finite modes are calculated.

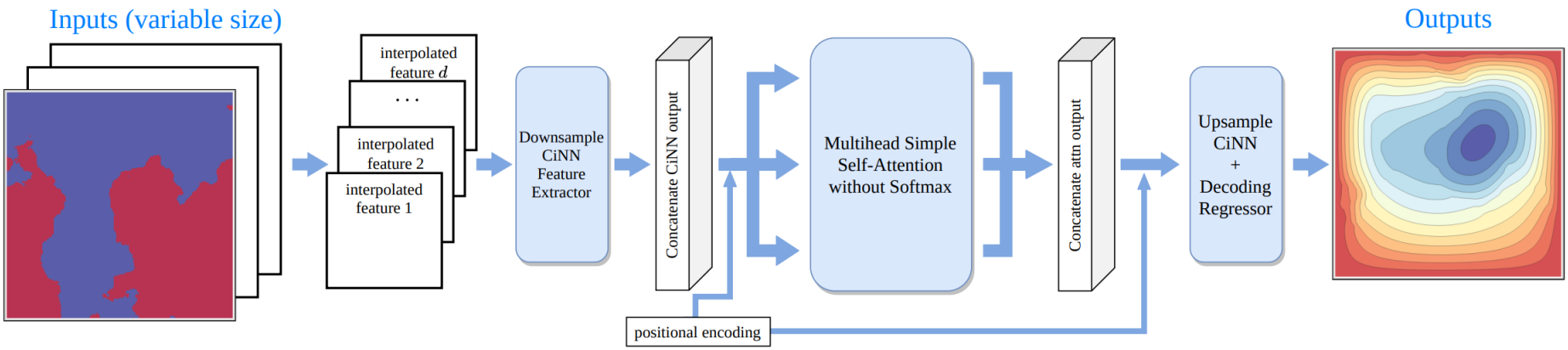


[6]Li, Zongyi, et al. "Fourier neural operator for parametric partial differential equations." arXiv preprint arXiv:2010.08895 (2020).

Contents

Proposed method

An attention-based operator learner on $\Omega \subset \mathbb{R}^2$. It utilizes CNN for feature extraction and decoding.



Simple Self-attention Encoder

The core module of the model is the **simple** self-attention encoder

$$y = y + \sigma(y + \text{Attn}_\dagger(y)), y \in \mathbb{R}^{n \times d}$$

where $\dagger \in \{\text{f}, \text{g}\}$:

- Fourier-type attention: $z = \text{Attn}_\text{f} := (\tilde{Q}\tilde{K}^T)V/n$,
- Galerkin-type attention: $z = \text{Attn}_\text{g} : Q(\tilde{K}^T\tilde{V})/n$,

and $\tilde{\diamond}$ denotes the layer normalization[7]. Generally, the function of it is to prevent elements and their gradients from exploding or vanishing.

[7] Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer normalization." arXiv preprint arXiv:1607.06450 (2016).

Important Analogy

The similar structure between the inner product and integration (inner product of two functions).

$$u_h = (u(x_1), ..., u(x_n)), v_h = (v(x_1), ..., v(x_n))$$

$$u_h \cdot v_h / n = \sum_{i=1}^{n} u(x_i) v(x_i) / n \sim \int uv dx, \text{as } n \to \infty$$

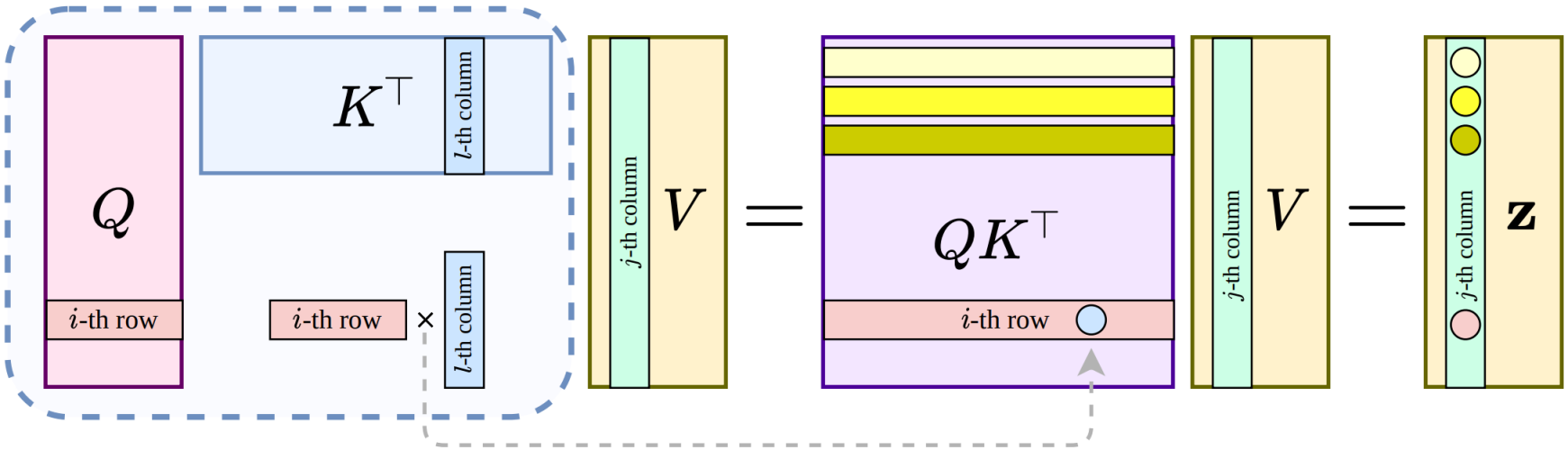
In this way, we can interpret the matrix multiplication as a batch of integrations.

From Matrix to Functions

Denote $\mathbf{q}(\cdot)$ as the function induced by Q: the value of $\mathbf{q}(\cdot)$ at the position x_i is the vector $(Q_{i,j})_{j=1}^{d}$, $i = 1, \ldots n$. Similarly, denote $\mathbf{k}(x_i) = (K_{i,j})_{j=1}^{d}$ and $\mathbf{v}(x_i) = (V_{i,j})_{j=1}^{d}$. Therefore,

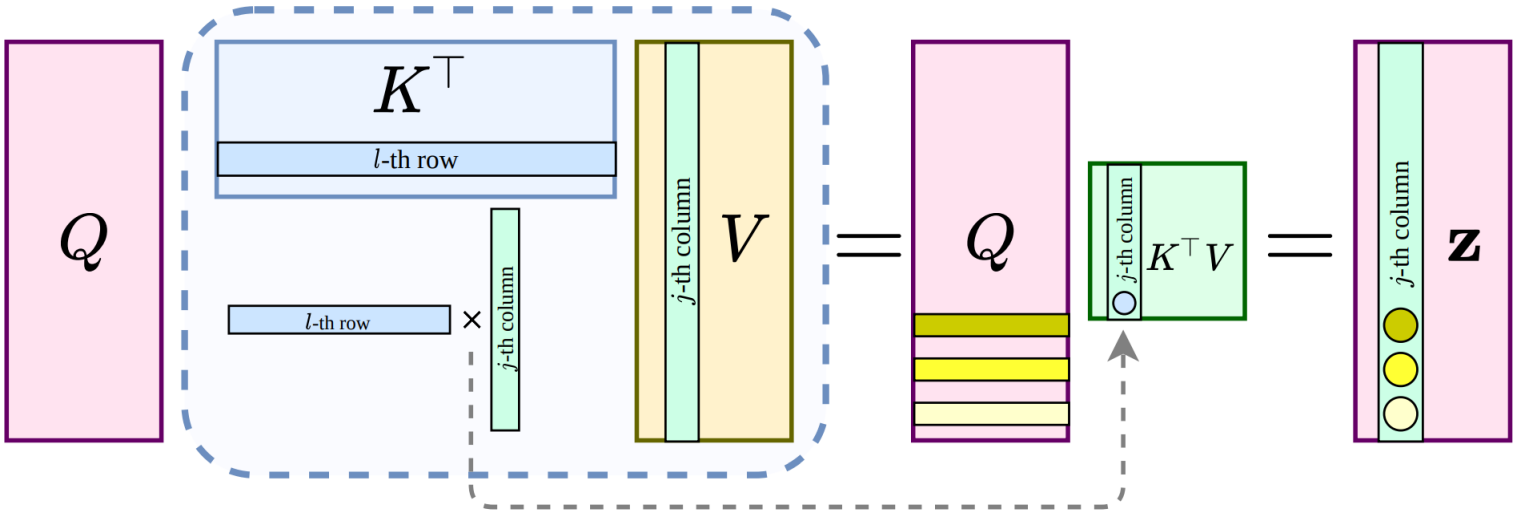
$$(QK^T)_{i,j} = \sum_{l=1}^{d} Q_{i,l} K_{j,l} = \mathbf{q}(x_i) \cdot \mathbf{k}(x_j)$$

Fourier-type Attention



$$\int (\mathbf{q}(x_i) \cdot \mathbf{k}(y))\mathbf{v}(y)dy \sim (QK^T V/n)_{i,\cdot} = \mathbf{z}_{i,\cdot} \sim \mathbf{z}(x_i)$$

Galerkin-type Attention



$$\sum_{l=1}^{d} \int \mathbf{k}_l(y)\mathbf{v}_j(y)dy\mathbf{q}_l(x) \sim (QK^TV/n)_{\cdot,j} = \mathbf{z}_{\cdot,j} \sim \mathbf{z}_j(x)$$

Galerkin-type Attention

$$\sum_{l=1}^{d} < \mathbf{k}_l(y), \mathbf{v}_j(y) > \mathbf{q}_l(x) = \mathbf{z}_j(x), j = 1, ..., d; x \in \Omega$$

compared to

$$\sum_{j=1}^{n} a(\phi_j, \phi_i) u_j =< f, \phi_i >, i = 1, ..., n$$

This indicates that the forward propagation has the similar structure as a parameterized Galerkin equation.

Contents

Céa Lemma for the Attention Operator

$$\min_{\theta} \|f - g_\theta(\mathbf{y})\|_{\mathcal{H}} \leq c^{-1} \min_{q \in \mathbb{Q}_h} \max_{v \in \mathbb{V}_h} \frac{|\mathfrak{b}(f_h - q, v)|}{\|v\|_{\mathcal{H}}} + \|f - f_h\|_{\mathcal{H}}$$

- Under some condition (known as Ladyzhenskaya–Babuška–Brezzi (LBB) condition), c is n-independent. In this case, g_θ has the capacity to approximate target T independent of the mesh size h.

Contents

Baseline Methods

ST: the standard softmax normalized scaled dot-product attention.
FNO: Fourier Neural Operator
LT: Efficient attention[8]

$$\mathbf{E}(Q, K, V) = \rho_q(Q)(\rho_k(K)^T V),$$

where ρ_q and ρ_k have two options:

$$\text{Scaling: } \rho_q(Y) = \rho_k(Y) = \frac{Y}{\sqrt{n}} \tag{1}$$

$$\text{Softmax: } \rho_q(Y) = \text{Softmax}_{\text{row}}(Y), \rho_k(Y) = \text{Softmax}_{\text{col}}(Y) \tag{2}$$

[8] Shen, Zhuoran, et al. "Efficient attention: Attention with linear complexities." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2021.

Example 1: Viscous Burgers' Equation

$$\begin{cases} \partial_t u + u\partial_x u = \nu \partial_{xx} u, (x,t) \in (0,1) \times (0,1] \\ u(x,0) = u_0(x), x \in (0,1) \end{cases}$$

The operator to be learned is

$$T : u_0(\cdot) \mapsto u(\cdot, 1)(\cdot)$$

Table 2: Evaluation relative error ($\times 10^{-3}$) of Burgers' equation 4.1.

	$n = 512$ $(b = 8)$	$n = 2048$ $(b = 8)$	$n = 8192$ $(b = 4)$
FNO1d [49]	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

Example 1: Viscous Burgers' Equation

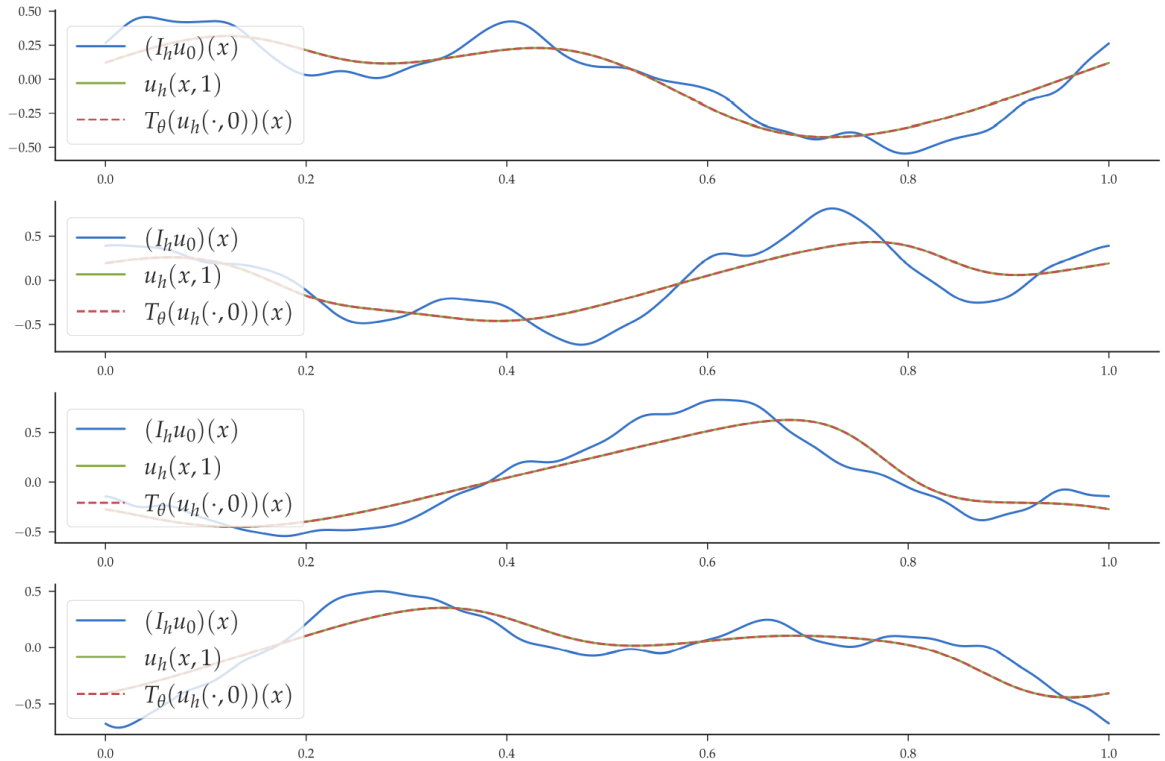


Figure 8: Evaluation results for 4 randomly chosen samples in the test set; the average relative error $= 1.079 \times 10^{-3}$.

Example 2: Darcy Flow

$$\begin{cases} -\bigtriangledown \cdot (a \bigtriangledown u) = f, \text{in} \Omega \\ u = 0, \text{on} \quad \partial\Omega \end{cases}$$

The operator to be learned is

$$T : a \mapsto u$$

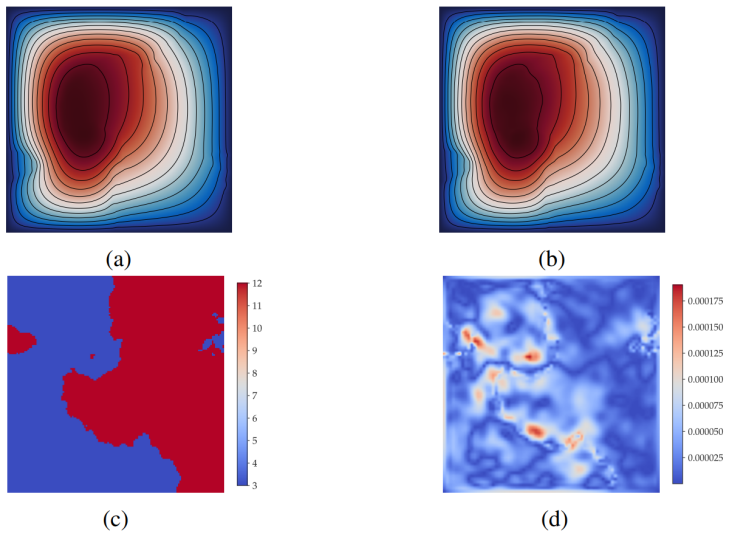


Figure 9: Interface Darcy flow in example 4.2: a randomly chosen sample from the test dataset. (a) the target being the finite difference approximation to the solution on a very fine grid; (b) the inference approximation by model evaluation (relative L^2-error 6.454×10^{-3}); (c) the input $a(x)$; (d) the L^∞-error distribution for the inference solution.

Example 2: Darcy Flow

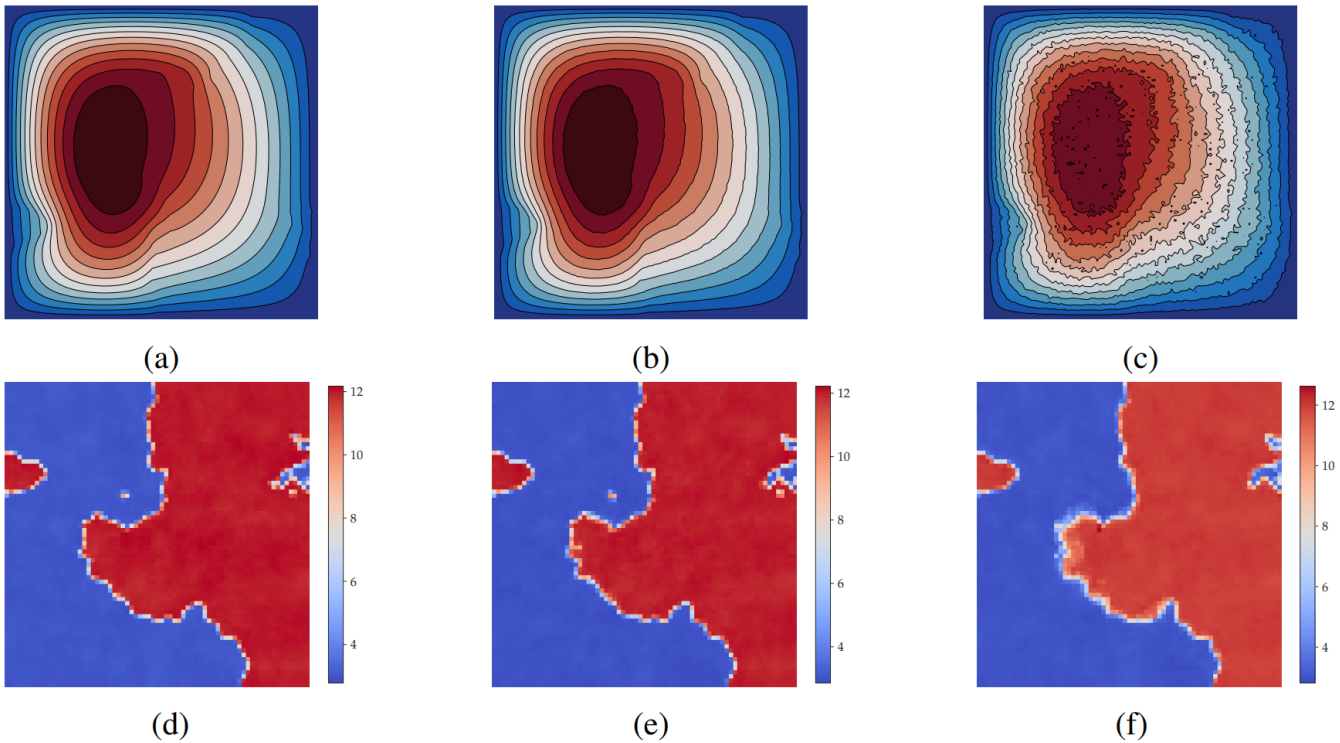
Table 3: Evaluation relative error ($\times 10^{-2}$) of Darcy interface problem 4.2.

	$n_f = 141, n_c = 43$	$n_f = 211, n_c = 61$
FNO2d [49]	1.09	1.09
FNO2d `1cycle` (only n_f)	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	0.919	0.935
GT Ln on K, V	**0.839**	0.900
ST Ln on Q, K	0.946	0.959
LT Ln on K, V	0.875	0.970

Example 3: Inverse Problem for Darcy Flow

The operator to be learned is

$$T : u_h \mapsto a$$



Example 3: Inverse Problem for Darcy Flow

Table 4: Evaluation relative error ($\times 10^{-2}$) of the inverse problem 4.3.

	$n_f = 141, n_c = 36$			$n_f = 211, n_c = 71$		
	$\epsilon = 0$	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0$	$\epsilon = 0.01$	$\epsilon = 0.1$
FNO2d (only n_f)	13.71	13.78	15.12	13.93	13.96	15.04
FNO2d (only n_c)	14.17	14.31	17.30	13.60	13.69	16.04
FT regular Ln	**1.779**	**2.467**	6.814	1.563	2.704	8.110
GT regular Ln	2.026	2.536	**6.659**	1.732	2.775	8.024
ST regular Ln	2.434	3.106	7.431	2.069	3.365	8.918
LT regular Ln	2.254	3.194	9.056	2.063	3.544	9.874
FT Ln on Q, K	1.921	2.717	6.725	**1.534**	**2.691**	8.286
GT Ln on K, V	1.944	2.552	6.689	1.799	2.764	**7.903**
ST Ln on Q, K	2.160	2.807	6.995	1.889	3.123	8.788
LT Ln on K, V	2.360	3.196	8.656	2.136	3.539	9.622

Contents

Strengths

This work

- theoretically proves the approximation property of the Transformer,
- empirically shows the proposed attention-based PDE solver outperforms its competitors,
- demonstrates the structural similarity between the Transformer and classical PDE solver.

Weakness

This work

- does not provide convincing interpretation about why to apply layer normalization to the Transformer,
- does not show more direct connections from the structural similarity to the approximation property of the Transformer.

Thank you