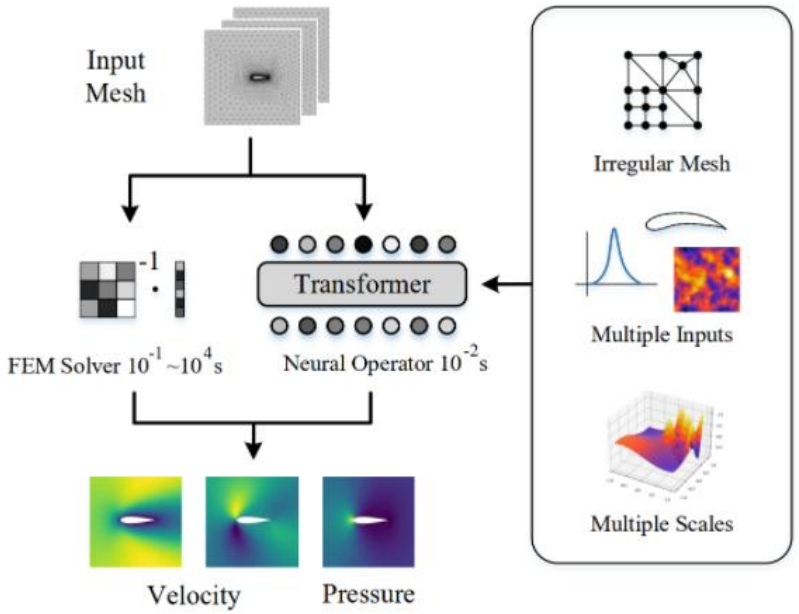
南京航空航天大學
NANJING UNIVERSITY OF

GNOT: A General Neural Operator Transformer for Operator Learning

Zhongkai Hao [1 2] Zhengyi Wang [1 3] Hang Su [1] Chengyang Ying [1] Yinpeng Dong [1 3]
Songming Liu [1] Ze Cheng [4] Jian Song [2] Jun Zhu [1 3]

ICML 2023

Background

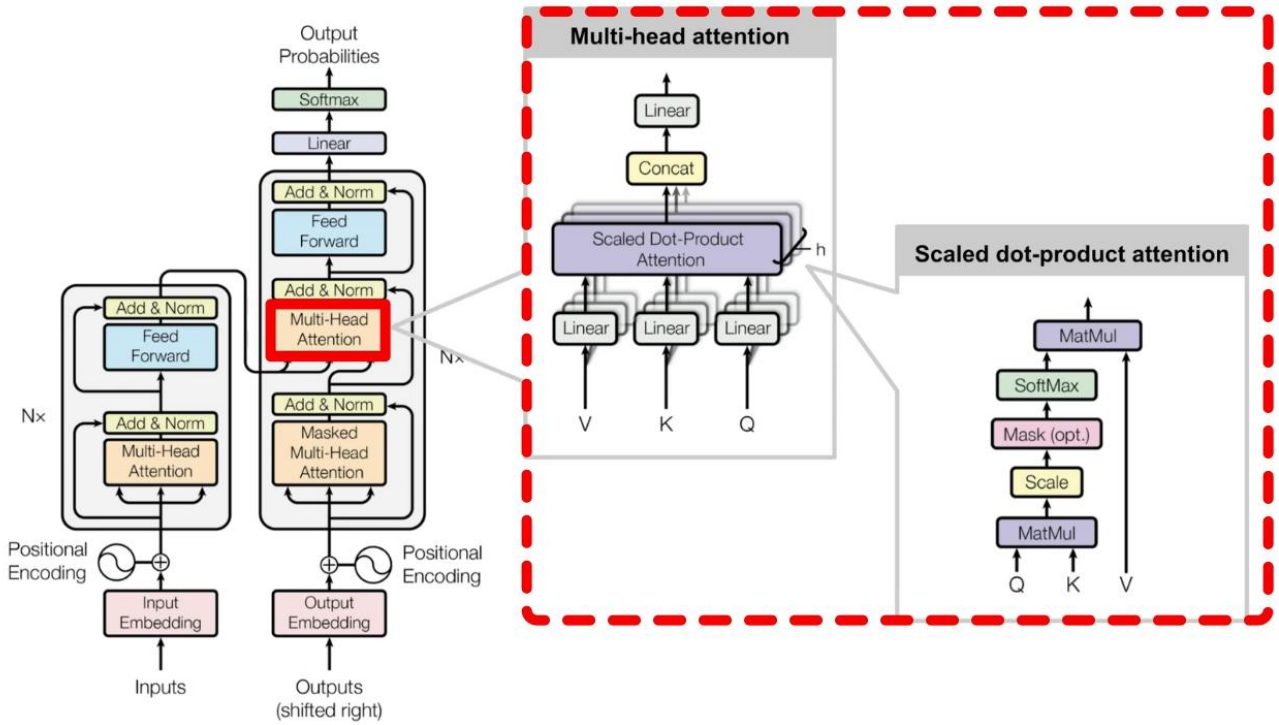


PDEs are usually solved by numerical methods like the finite element method (FEM). It is often computationally expensive for high dimensional problems.Besides, machine learning methods (Lu et al., 2019; Li et al., 2020; 2022b) are proposed to accelerate solving PDEs by learning an operator mapping from the input functions to the solutions of PDEs.

Operator learning for practical real-world problems is highly challenging and the performance can be unsatisfactory. There are several major challenges in current methods: irregular mesh, multiple inputs, and multi-scale problems.

Background

Transformer：

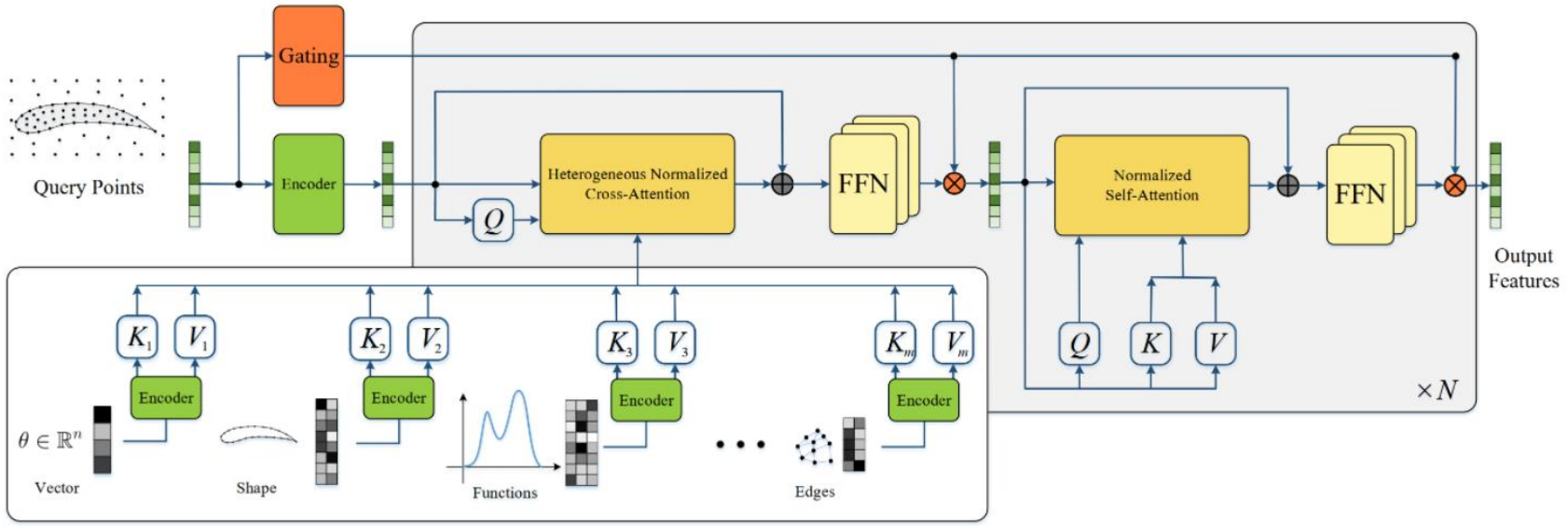


$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

- $Q \in \mathbb{R}^{N \times d_k}$：查询矩阵（Query），由查询序列 X_q 经过投影得到。
- $K \in \mathbb{R}^{M \times d_k}$：键矩阵（Key），由输入序列 X_k 经过投影得到。
- $V \in \mathbb{R}^{M \times d_v}$：值矩阵（Value），由输入序列 X_v 经过投影得到。
- d_k：键向量的维度。
- N：查询序列的长度（查询点个数）。
- M：输入序列的长度（键/值的个数）。

Method

GNOT:



Overview of the model architecture. First, we encode input query points and input functions with different MLPs. Then we update features of query points using a heterogenous normalized cross-attention layer and a normalized self-attention layer. We use a gate network using geometric coordinates of query points to compute a weighted average of multiple expert FFNs. We output the features after processing them using N layers of the attention block.

Method

We consider PDEs in the domain $\Omega \subset R^d$ and the function space $\mathcal{H}$ over Ω, including boundary shapes and source functions. Our goal is to learn an operator $\mathcal{G}$ from the input function space $\mathcal{A}$ to the solution space $\mathcal{H}$.

$$\mathcal{G}: \mathcal{A} \to \mathcal{H}$$

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

$$\mathcal{D} = \{(\alpha_k, u_k)\}_{1 \le k \le D}, u_k = \mathcal{G}(\alpha_k).$$

$$\mathcal{A}_k = \{(x_i^j, \alpha_k^{i,j})\}_{\substack{1 \le j \le m \\ 1 \le i \le N_j}} \cup \theta_k, \alpha_k^{i,j} = \alpha_k^j(x_i^j). \qquad \{(y_i, u_k^i)\}_{1 \le i \le N'}, u_k^i = u_k(y_i).$$

$$\min_{w \in W} \frac{1}{D} \sum_{k=1}^{D} \frac{1}{N'} \left\| \widetilde{\mathcal{G}}_w(\mathcal{A}_k) - \{u_k^i\}_{1 \le i \le N'} \right\|_2^2$$

Method

General Input Encoding

• Parameter vector $\theta \in R^P$: We could directly encodethe parameter vector using the MLP, i.e, $Y = f_w(\theta)$and $Y \in R^{1\times n_e}$.

• Boundary shape $\{x_i\}_{1\le i\le N}$: If the solution relies on the shape of the boundary, we propose to extract all these boundary points as input function and embed the position of these points with MLP. Specifically, $Y = (f_w(x_i))_{1\le i\le N} \in R^{Nd}$.

• Domain distributed functions $\{(x_i, \alpha_i)\}_{1\le i\le N}$: If the input function is distributed over a domain or a mesh,we need to encode both the position of nodes and the function values, i.e. $Y = (f_w(x_i, \alpha_i))_{1\le i\le N} \in R^{Nd}$.

• Domain knowledge: For example, we could encode the extra features of mesh points $\{(x_i, z_i)\}_{1\le i\le N}$ and edge information of the mesh $\{(x_i^{src}, x_i^{dst}, e_i)\}_{1\le i\le N}$. This extra information is usually generated when collecting the data by solving FEMs. We use MLPs to encode them into $Y = (f_w(x_i, z_i))_{1\le i\le N}$ and $Y = (f_w(x_i, z_i))_{1\le i\le N}$.

Method

Heterogeneous Normalized Attention Block

The attention is computed as follows:

$$z_t = \sum_i \frac{exp(q_t \cdot k_i/\tau)}{\sum_j exp(q_t \cdot k_j/\tau)} v_i$$

For self-attention models, q, k, v are obtained by applying a linear transformation to input sequence $\mathrm{X} = (x_i)_{1 \le i \le N}$, i.e, $q_i = w_q x_i, k_i = w_k x_i, v_i = w_v x_i$.

For cross attention models, q comes from the query sequence X while keys and values come from another sequence $\mathrm{Y} = (y_i)_{1 \le i \le M}$, i.e, $q_i = w_q x_i, k_i = w_k y_i, v_i = w_v y_i$.

The computational cost of the attention is $O(N^2 n_e)$ for self attention and $O(\mathrm{NM}n_e)$ for cross attention where n_e is the dimension of embedding.

Method

Heterogeneous Normalized Attention Block

Here we propose a novel attention layer with a linear computational cost that could handle long sequences. We first normalize these sequences respectively.

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

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

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

$$z_t = \sum_i \frac{\tilde{q}_t \cdot \tilde{k}_i}{\sum_j \tilde{q}_t \cdot \tilde{k}_j} \cdot v_i \qquad \alpha_t = \left(\sum_j \tilde{q}_t \cdot \tilde{k}_j\right)^{-1} \qquad z_t = \sum_i \alpha_t (\tilde{q}_t \cdot \tilde{k}_i) \cdot v_i = \alpha_t \tilde{q}_t \cdot \left(\sum_i \tilde{k}_i \otimes v_i\right).$$

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

Method

Heterogeneous Normalized Attention Block

Then we compute the cross-attention as follows,

$$z_t = \tilde{q}_t + \frac{1}{L}\sum_{l=1}^{L}\sum_{i_l=1}^{N_l} \alpha_t^l\,(\tilde{q}_t \cdot \tilde{k}_{i_l}) \cdot v_{i_l} = \tilde{q}_t + \frac{1}{L}\sum_{l=1}^{L} \alpha_t^l \tilde{q}_t \cdot (\sum_{i_l=1}^{N_l} \tilde{k}_{i_l} \otimes v_{i_l}).$$

$$where\ \alpha_t^l = \frac{1}{\sum_{j=1}^{N_l} \tilde{q}_t \cdot \tilde{k}_j}$$

We see that the cross-attention aggregates all information from input functions and extra information. We also add an identity mapping as skip connection to ensure the information is not lost. The computational complexity of it is $O((N + \sum_l N_l)n_e{}^2)$ also linear with sequence length.

Method

Heterogeneous Normalized Attention Block

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

$$z_t^{'} = \sum_i \alpha_t(\tilde{q}_t \cdot \tilde{k}_i) \cdot v_i$$

where all of q, k and v are computed with the embedding z_t as

$$q_t = W_q \hat{z}_t, k_t = W_k \hat{z}_t, v_t = W_v \hat{z}_t.$$

We use the cascade of a cross-attention layer and a selfattention layer as the basic block of our model. We tile multiple layers and multiple heads similar to other transformer models. The embedding z_t and $z_t^{'}$ are divided into H heads as $z_t = Concat(Z_t^i)_{i=1}^H$ and $z_t^{'} = Concat(Z_t^{'i})_{i=1}^H$. Each head Z_t^i can be updated using the above Eq.

Method

Geometric Gating Mechanism

We design a geometric gating network that inputs the coordinates of the query points and outputs unnormalized scores $G_i(x)$ for averaging expert networks. In each layer of our model, we use K subnetworks for the MLP denoted by $E_i(\cdot)$. The update of z_t and $z_t^{'}$ in the feedforward layer after the above Eq. is replaced by the following equation when we have multiple expert networks as

$$z_t \longleftarrow z_t + \sum_{i=1}^{K} p_i(x_t) \cdot E_i(z_t).$$

The weights for averaging the expert networks are computed as

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

where the gating network $G(\cdot) : R^d \rightarrow R^k$ 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.

Experiment

We list the challenges of these datasets in Table 1 where “A”, “B”, and “C” represent the problem has irregular mesh, has multiple input functions, and is multi-scale, respectively.

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**

Table 1. Our main results of operator learning on several datasets from multiple areas. The types like u, v are the physical quantities to predict and types like ”part“ denotes the size of the dataset. ”-“ means that the method is not able to handle this dataset. Lower scores mean better performance and the best results are **bolded**.

Experiment

南京航空航天大學 NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

Scaling Experiments

One of the most important advantages of transformers is that its performance consistently gains with the growth of the number of data and model parameters. Here we conduct a scaling experiment to show how the prediction error varies when the amount of data increases. We use the **NS2d-c** dataset and predict the pressure field p. We choose **MIONet** as the baseline and the results are shown in Fig 3.

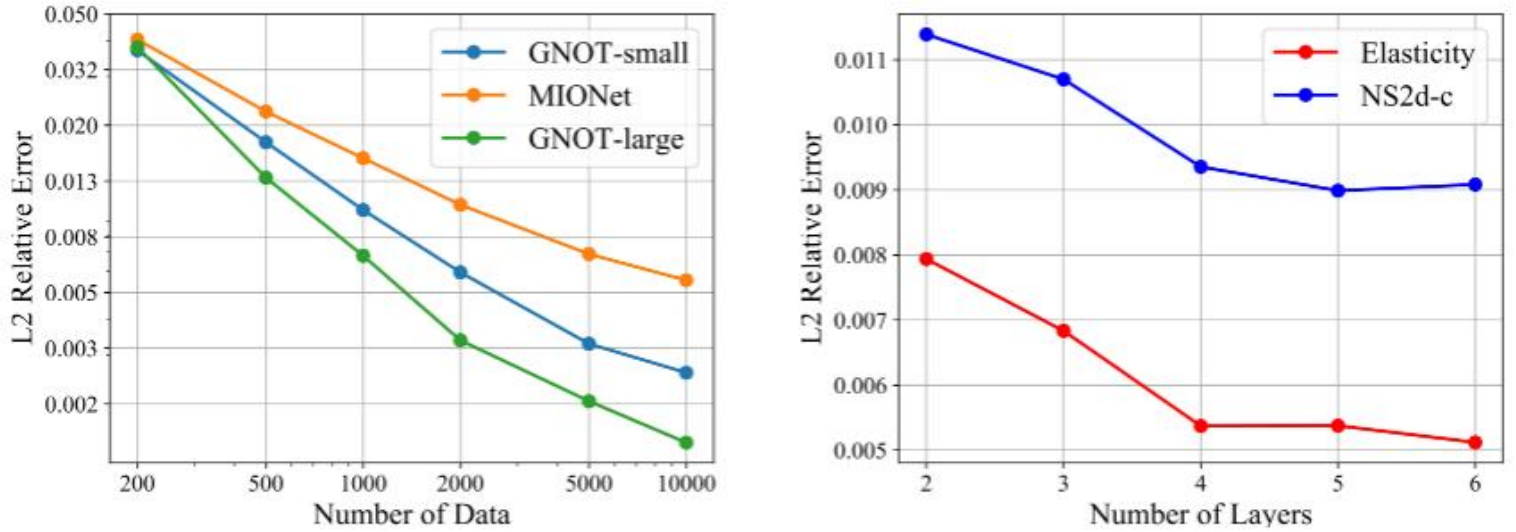


Figure 3. Results of scaling experiments for different dataset sizes (left) and different numbers of layers (right).

Experiment

南京航空航天大學
NANJING UNIVERSITY OF AERONAUTICS AND ASTRONAUTICS

Ablation Experiments

	NACA	Elasticity	NS2d-c (p)
cross + cross	3.52e-2	3.31e-2	1.50e-2
self + cross	9.53e-3	1.25e-2	9.89e-2
cross + self	**7.57e-3**	**8.65e-3**	**7.41e-3**

Table 2. Experimental results for the necessity and order of different attention blocks.

N_{experts}	error	N_{heads}	error
1	0.04212	1	0.04131
3	0.03695	4	0.04180
8	0.04732	8	0.04068
16	0.04628	16	0.03952

Table 3. Results for ablation experiments on the influence of numbers of experts N_{experts} (left two columns) and numbers of attention heads N_{heads} (right two columns).

Thanks