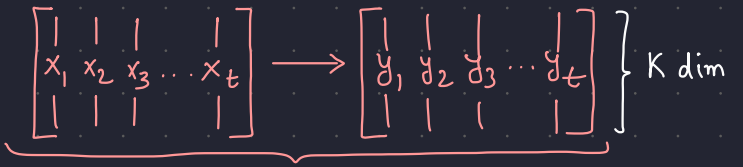
A General Neural Operator Transformer for Operator Learning

Self Attention

- Self Attention is just a sequence to sequence operation:



Weighted Avg over all input vectors $y_i = \sum_j w_{ij} x_j$

- Weights are not learnable parameters: $\underbrace{w'_{ij} = x_i^T x_j}_{-\infty \text{ to } \infty} \Rightarrow \underbrace{w_{ij} = \text{softmax}(w'_{ij})}_{0 \text{ to } 1}$

- Three things happening here:
  - → Each x_i is compared to every other vector to establish weights for its own output y_i
  - → Each x_i is compared to every other vector to establish weights for other outputs y_j
  - → Each x_i is used as a part of weighted sum to get each output vector

- Can we somehow come up with a clever way to get these 3 things done?

Self Attention

- Self Attention is just a sequence to sequence operation:

$$\underbrace{\begin{bmatrix} | & | & | & & | \\ x_1 & x_2 & x_3 & \cdots & x_t \\ | & | & | & & | \end{bmatrix} \longrightarrow \begin{bmatrix} | & | & | & & | \\ y_1 & y_2 & y_3 & \cdots & y_t \\ | & | & | & & | \end{bmatrix}}_{\text{K dim}}$$

Weighted Avg over all input vectors $y_i = \sum_j w_{ij} x_j$

- Weights are not learnable parameters: $\underbrace{w'_{ij} = x_i^T x_j}_{-\infty \text{ to } \infty} \Rightarrow \underbrace{w_{ij} = \text{softmax}(w'_{ij})}_{0 \text{ to } 1}$

- Three things happening here:
  - → Each x_i is compared to every other vector to establish weights for its own output y_i : $q_i = \overbrace{W_q}^{\text{Neural Nets}} x_i$
  - → Each x_i is compared to every other vector to establish weights for other outputs y_j : $k_i = W_k x_i$
  - → Each x_i is used as a part of weighted sum to get each output vector : $v_i = W_v x_i$

$$w'_{ij} = q_i^T k_j \Rightarrow w'_{ij} = \frac{q_i^T k_j}{\sqrt{K}}$$

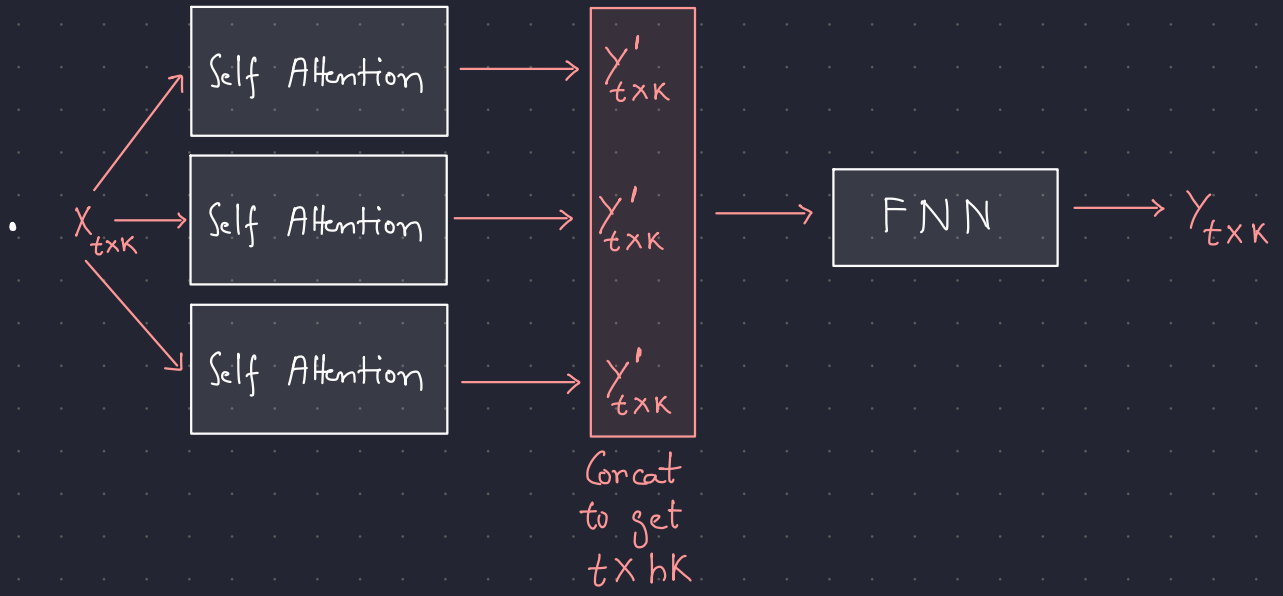
Scaling dot product for stability

$\sqrt{K}$ ↓ Length

$$w_{ij} = \text{softmax}(w'_{ij})$$

$$y_i = \sum_j w_{ij} v_j$$

Multi Head Attention



GNOT Architecture

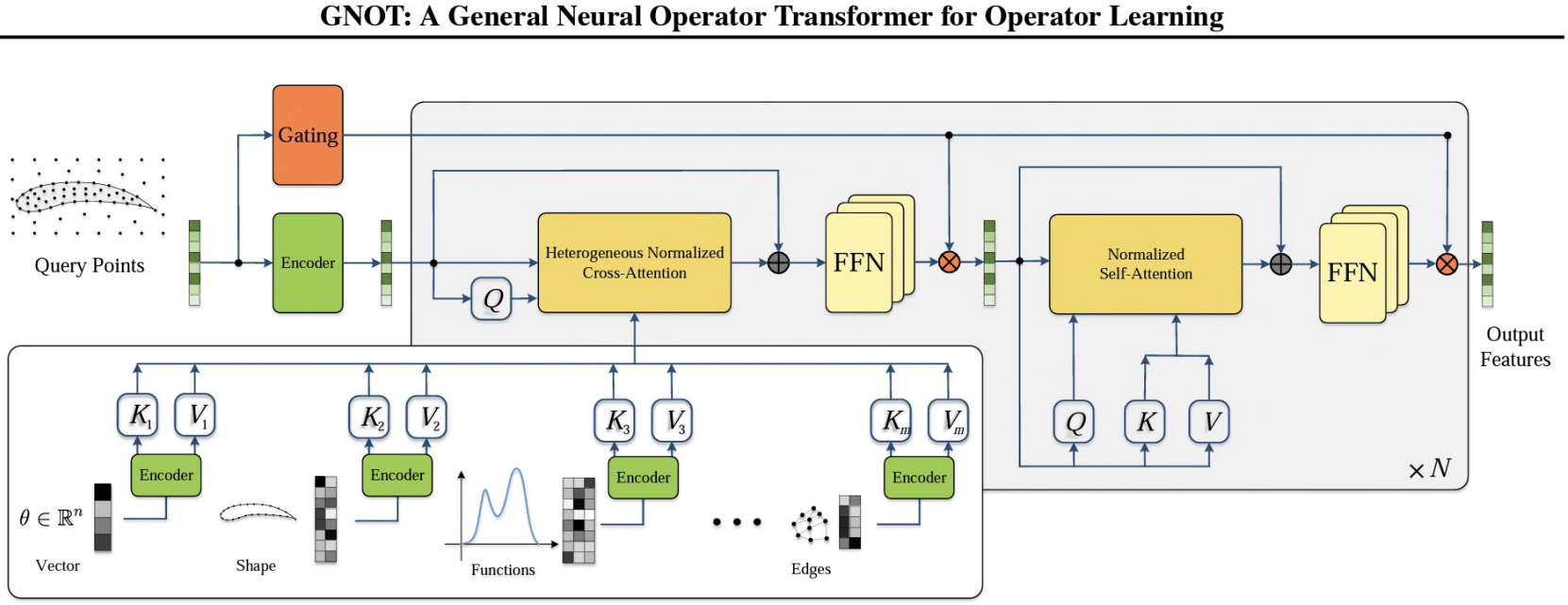


Figure 2. Overview of the model architecture. First, we encode input query points and input functions with different MLPs. Then we update features of query points using a heterogenous normalized cross-attention layer and a normalized self-attention layer. We use a gate network using geometric coordinates 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.

GNOT

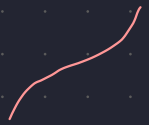
Problem Formulation

- Query Points x_i

- Input function u^0

- Boundary Shape x_i'

- Final Solution u



Task is to map $u^0 \longrightarrow u$ using domain Knowledge x_i'

$\Longrightarrow$

Mean Squared Error $\left[G(A), u \right]$

(u^0, x_i')

GNOT

General Input Encoding

- Query Points x_i  $1 < i < N_q$
- Input function u°  $(x_i, u_i^\circ)\ ;\ i < N_i$
- Boundary Shape x_i'  $1 < i < N_b$

} All different dimensions

General Input Encoding resolves this issue

- $x_i \xrightarrow{\text{NN}} q$ $\quad 1 \times n_e$, $N_q \times n_e$
- $(x_i, u_i^\circ) \xrightarrow{\text{NN}} K$ $\quad 1 \times n_e$; $\xrightarrow{\text{NN}} V$
- $(x_i') \xrightarrow{\text{NN}} K$; $\xrightarrow{\text{NN}} V$

These NN are called Encoders

GNOT

Heterogeneous Normalised Attention

- From Encoder NNs we have $\{q_i\}_{i\le N}$ $\{K_i\}_{i\le M}$ $\{v_i\}_{i\le M}$ — $N\ n_e$ $M\ n_e$

- $$z_t = \sum_i \frac{\exp(q_t \cdot K_i/\tau)}{\sum_j \exp(q_t \cdot K_j/\tau)} v_i \Bigg\} O(NM\, n_e)$$

  n_e ↓ dimensions of Embeddings

- As PDE problems can have N and M in millions, it scales quite badly.

- Here is the proposed Modification:

$$\tilde{q}_i = \text{softmax}(q_i) = \frac{\exp(q_i)}{\sum_j \exp(q_{ij})} \; ; j \le n_e \qquad \tilde{K}_i = \text{softmax}(K_i) = \frac{\exp(K_i)}{\sum_j \exp(K_{ij})} \; ; j \le n_e$$

$$z_t = \sum_i \underbrace{\frac{\tilde{q}_t \cdot \tilde{K}_i}{\sum_j \tilde{q}_t \tilde{K}_i}}_{\text{Normalisation}} \cdot v_i = \alpha_t \sum_i (\tilde{q}_t \cdot \tilde{K}_i) \cdot v_i = \alpha_t\, \tilde{q}_t \cdot \underbrace{\left(\sum_i \tilde{K}_i \otimes v_i\right)}_{O(M n_e^2)}$$

$K_i : 1 \times n_e$, $v_i : 1 \times n_e$ → $\otimes$

$\overbrace{\quad\quad}^{O(N n_e^2)}$ $\Big\}\ O((M+N) n_e^2)$

- We have Queries, Keys and values from different embeddings.

$$z_t = \tilde{q}_t + \frac{1}{L}\sum_{l=1}^{L} \alpha_t^l \tilde{q}_t \cdot \left(\sum_i \tilde{K}_i^l \otimes V_i^l\right) \Bigg\} O\left(\left(N + \sum_l N_l\right) n_e^2\right)$$

Geometric Gating

- Inspired from domain decomposition techniques where each Net is made expert in a specific domain.

  Soft domain Decomposition

- Query Points coordinates $\xrightarrow[E_i()]{K \text{ diff } NN} G_i(x)$

- $$Z_t = Z_t + \sum_{i=1}^{K} P_i(x_t) \cdot E_i(Z_t)$$

  $$\downarrow$$

  $$\frac{\exp(G_i(x))}{\sum_{i=1}^{K} \exp(G_i(x))}$$

- We can learn Gating as NN or fix them with prior Knowledge
- Also used in XPINNs

Conclusions

GNOT: A General Neural Operator Transformer for Operator Learning

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**.