AI in the Sciences and Engineering HS 2025: Lecture 8

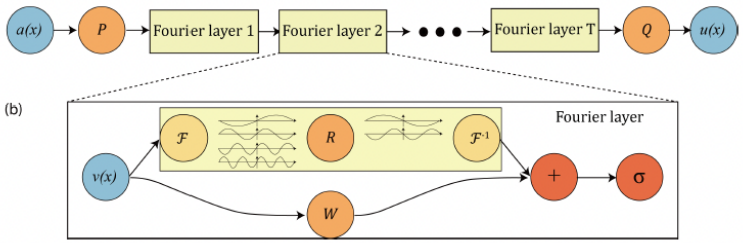
Siddhartha Mishra

Computational and Applied Mathematics Laboratory (CamLab)
Seminar for Applied Mathematics (SAM), D-MATH (and),
ETH AI Center (and) Swiss National AI Institute (SNAI) ,
ETH Zürich, Switzerland.

What we have learnt so far ?

- AIM: Learn PDEs using Deep Neural Networks
- Operator Learning: Learn the PDE Solution Operator from data.
- Examples: FNO, CNO.

Fourier Neural Operators: Li et al, 2020



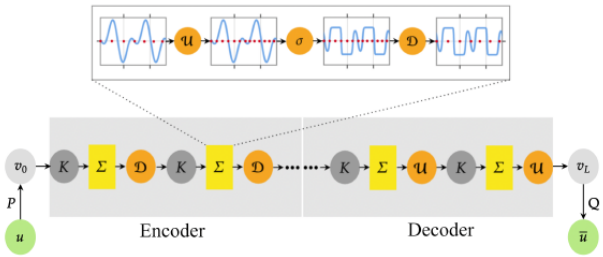
- Translation invariant Kernel $K(x, y) = K(x - y)$
- Use Fourier and Inverse Fourier Transform to define the KIO:

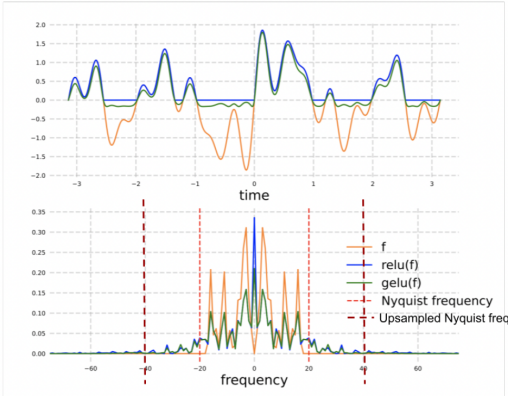
$$\int_D K(x - y)v(y)dy = \mathcal{F}^{-1}(\mathcal{F}(K)\mathcal{F}(v))(x)$$

- Parametrize Kernel in Fourier space.
- Nonlocal Convolutions + Local Activations: $[\mathcal{P}\sigma] = \sigma$
- Fast implementation through FFT
- Widely used+ Backed by solid Theory [1]

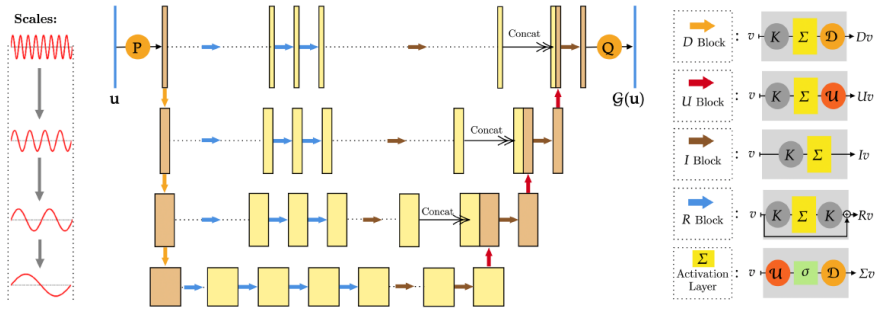
[1] Kovachki, Lanthaler, SM 2020, Lanthaler, Molinaro, Hadorn, SM 2022

Convolutional Neural Operators: Raonic, SM et al, 2023

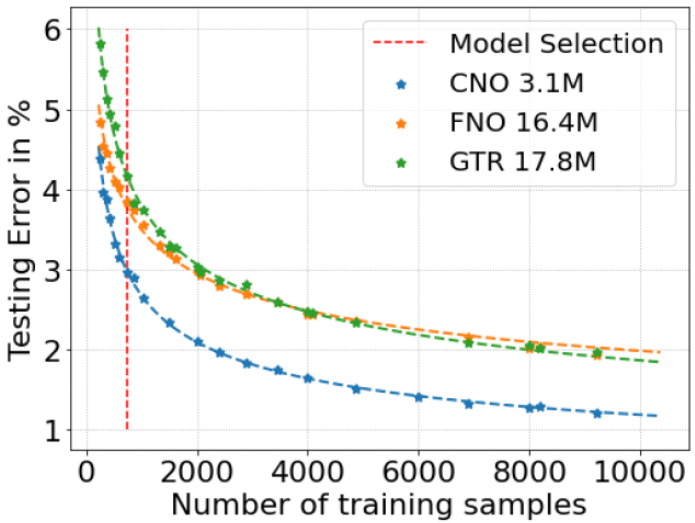


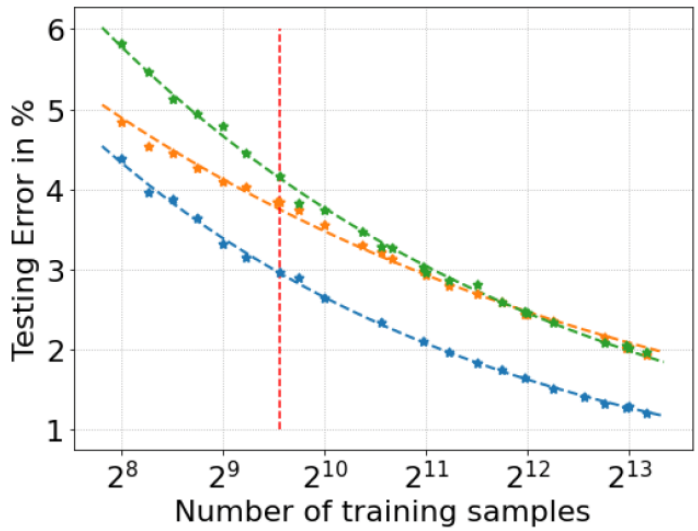


- I: Use Convolutional Kernels in Physical space
- II: Modulated Nonlocal activations for Alias-free processing.
- CNO instantiated as a modified Operator UNet

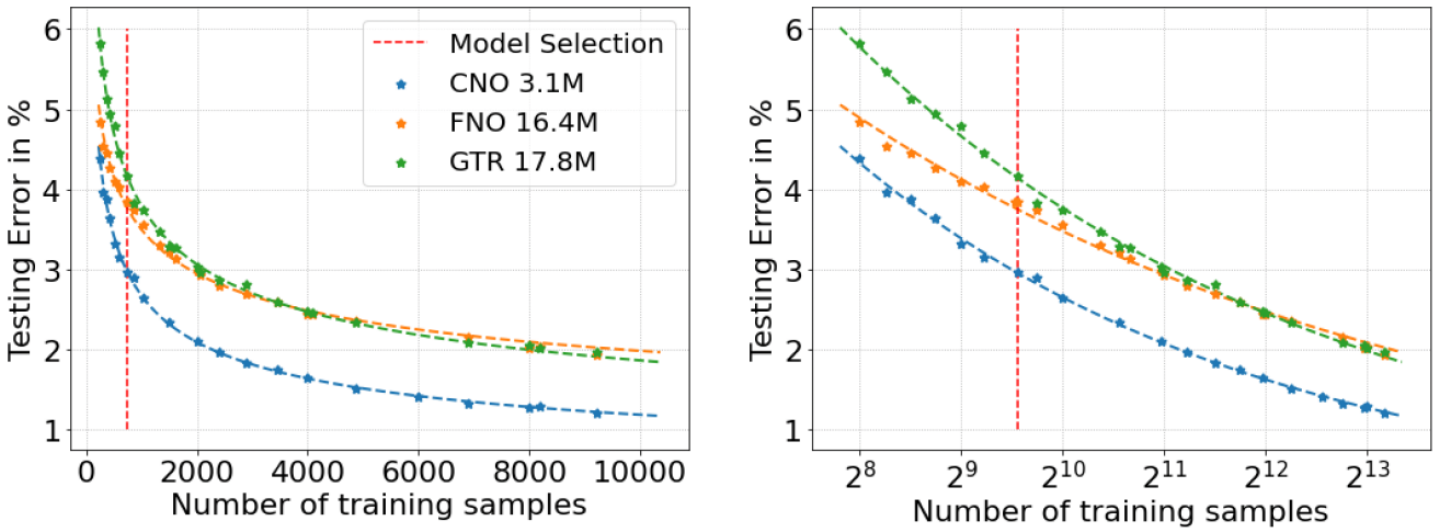


How does CNO/FNO Scale ?



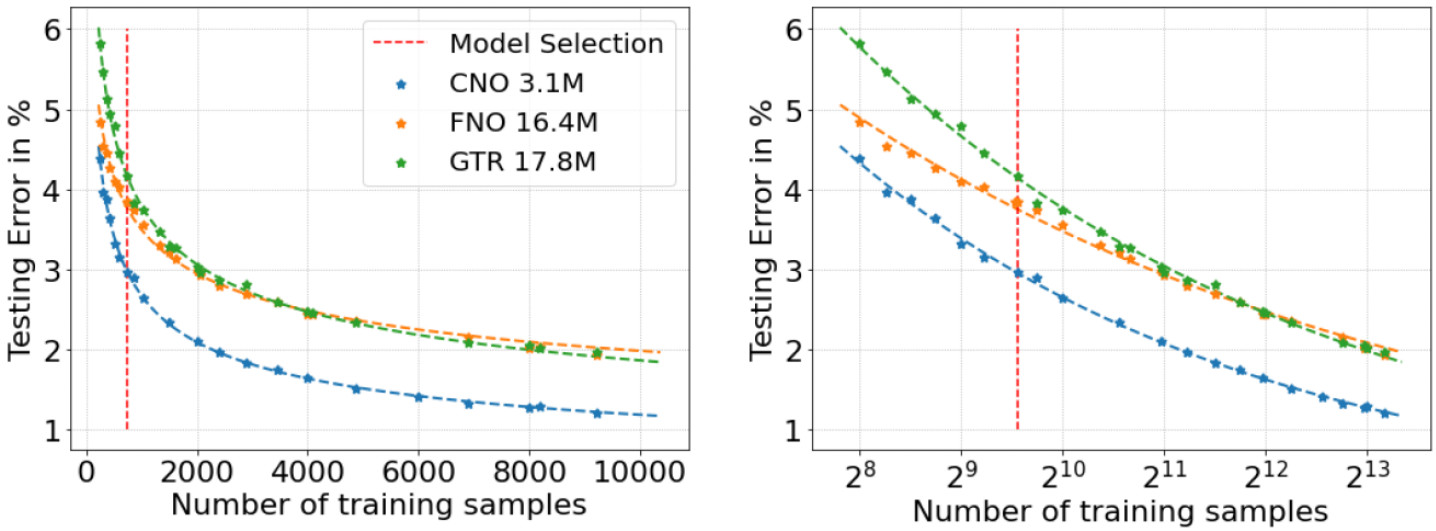


How does CNO/FNO Scale ?



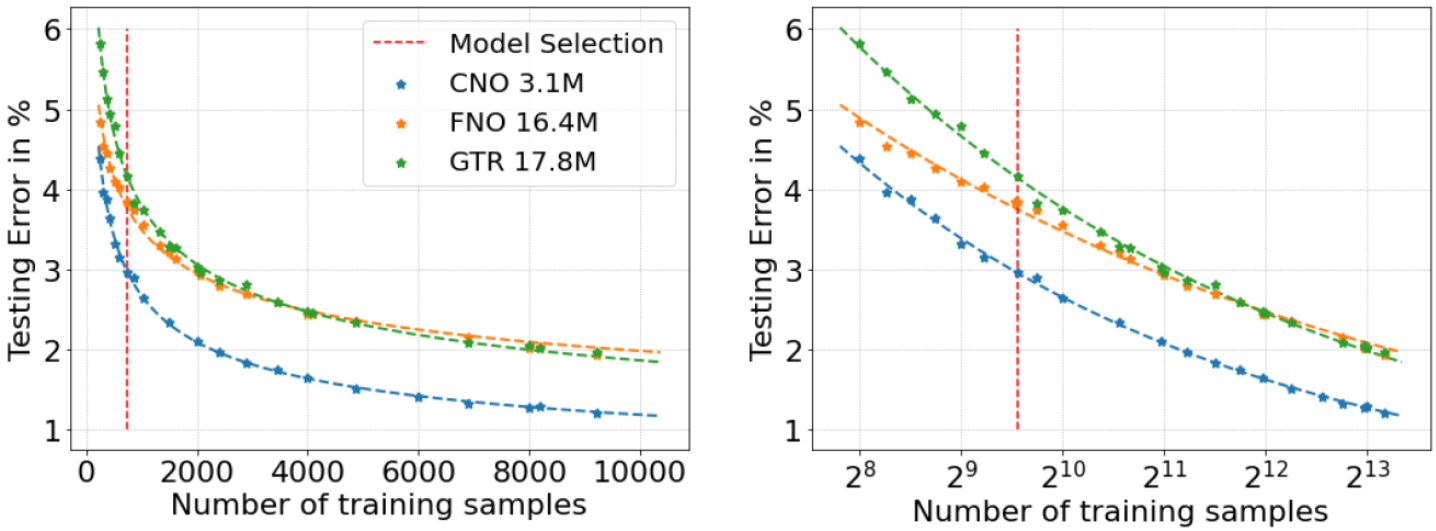
- Success is a curve, not a point !!

How does CNO/FNO Scale ?



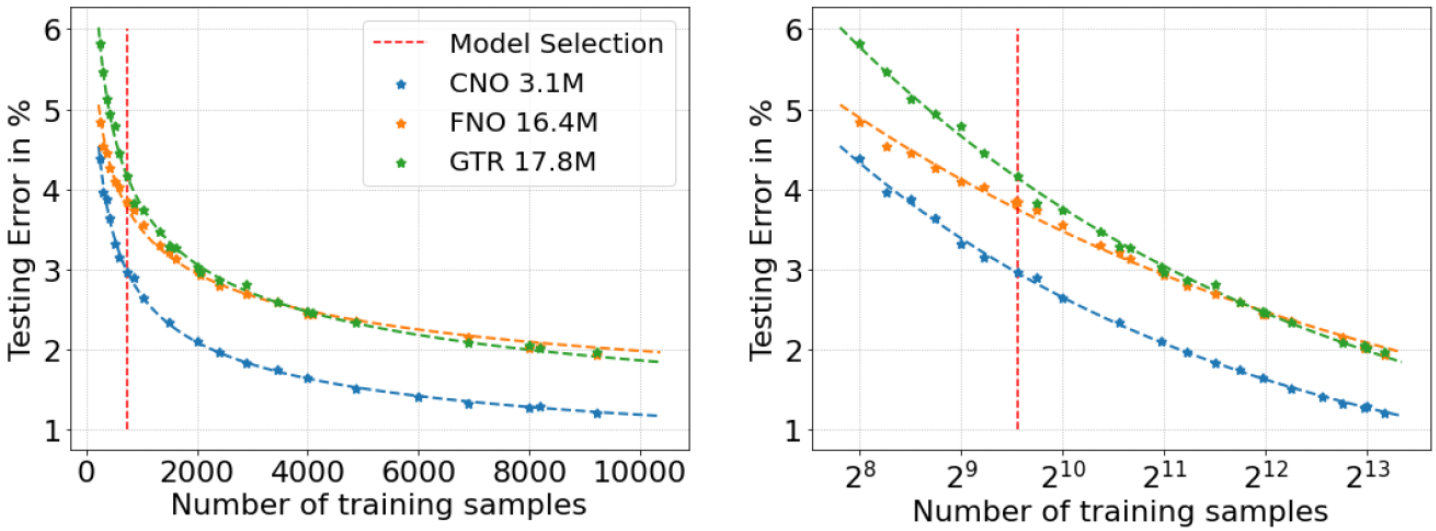
- Success is a curve, not a point !!
- Models Scale with sample size: $\mathcal{E} \sim N^{-\alpha}$ but with α small
- Theory: Lanthaler, SM, Karniadakis, De Ryck, SM,

How does CNO/FNO Scale ?



- Success is a curve, not a point !!
- Models Scale with sample size: $\mathcal{E} \sim N^{-\alpha}$ but with α small
- Theory: Lanthaler, SM, Karniadakis, De Ryck, SM,
- ML models require Big Data: $\mathcal{O}(10^3) - \mathcal{O}(10^4)$ training samples per Task
- Very Difficult to obtain Data for PDEs.

How does CNO/FNO Scale ?



- Success is a curve, not a point !!
- Models Scale with sample size: $\mathcal{E} \sim N^{-\alpha}$ but with α small
- Theory: Lanthaler, SM, Karniadakis, De Ryck, SM,
- ML models require Big Data: $\mathcal{O}(10^3) - \mathcal{O}(10^4)$ training samples per Task
- Very Difficult to obtain Data for PDEs.
- Can models Scale better ?

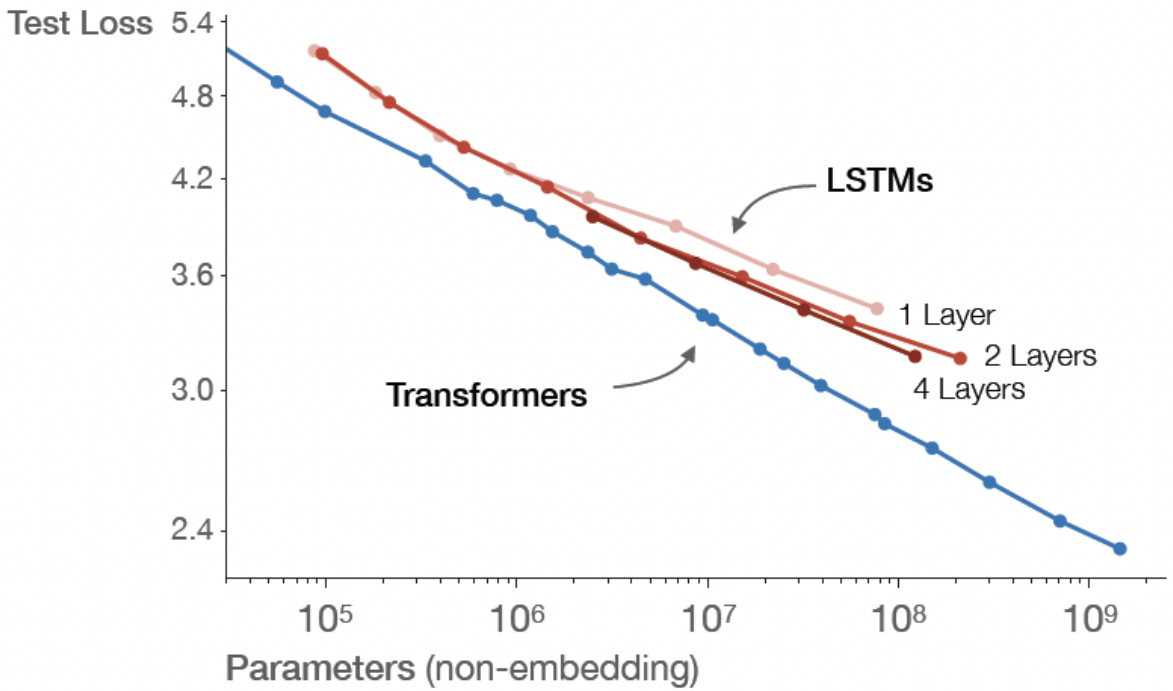
What Architectures Scale the best ?

What Architectures Scale the best ?

- What happens in Language Modeling ?
- Results of Kaplan et. al., 2020.

What Architectures Scale the best ?

- What happens in Language Modeling ?
- Results of Kaplan et. al., 2020.



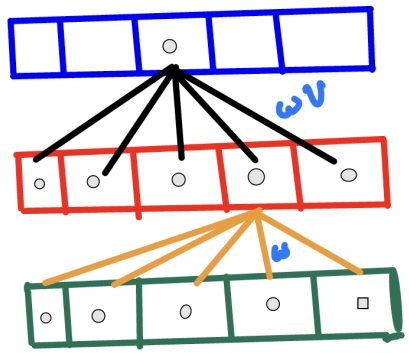
Move Towards Transformers

Attention is all you need !!

- Introduced by Vaswani et. al, 2017
- Given K-inputs $\mathsf{v} \in \mathbb{R}^{K\times n}$ of n-features:
- Self-Attention: $\mathbb{A} : \mathbb{R}^{K\times n} \mapsto \mathbb{R}^{K\times n}$ such that $\mathsf{u} = \mathbb{A}\mathsf{v}$:

$$\mathsf{u}_k = \mathsf{W}\sum_{j=1}^{K} \mathrm{softmax}_k\left(\frac{\langle \mathsf{Q}\mathsf{v}_k, \mathsf{K}\mathsf{v}_j\rangle}{\sqrt{m}}\right)\mathsf{V}\mathsf{v}_j, (\mathrm{softmax}\ (w))_i = \frac{e^{w_i}}{\sum_{\ell=1}^{L} e^{w_\ell}}$$

- Query $\mathsf{Q} \in \mathbb{R}^{m\times n}$, Key $\mathsf{K} \in \mathbb{R}^{m\times n}$, Value $\mathsf{V} \in \mathbb{R}^{m\times n}$.
- Output Matrix $\mathsf{W} \in \mathbb{R}^{n\times m}$



Multi-Head Attention

- Multi-Head Self-Attention:

$$\mathsf{u}_k = \sum_{h=1}^{H} \mathsf{W}_h \sum_{j=1}^{K} \mathrm{softmax}_k \left(\frac{\langle \mathsf{Q}_h \mathsf{v}_k, \mathsf{K}_h \mathsf{v}_j \rangle}{\sqrt{m}} \right) \mathsf{V}_h \mathsf{v}_j$$

Attention as a Neural Operator: I

- Let $\mathsf{v} \in C(D, \mathbb{R}^n)$ be the input function.
- $x_k \in D$ be sampling points on a Regular Grid with size Δ
- Apply Self-Attention to Tokens $\mathsf{v}_k = \mathsf{v}(x_k)$:

$$\mathsf{u}_k = \mathsf{W} \sum_{j=1}^{K} \mathrm{softmax}_k \left(\frac{\langle \mathsf{Q}\mathsf{v}_k, \mathsf{K}\mathsf{v}_j \rangle}{\sqrt{m}} \right) \mathsf{V}\mathsf{v}_j, (\mathrm{softmax}\ (w))_i = \frac{e^{w_i}}{\sum_{\ell=1}^{L} e^{w_\ell}}$$

- Passing to the limit as $\Delta \to 0$ yields

$$\mathsf{u}(x) = \mathbb{A}(\mathsf{v})(x) = \mathsf{W} \int_D \frac{e^{\frac{\langle \mathsf{Q}\mathsf{v}(x), \mathsf{K}\mathsf{v}(y) \rangle}{\sqrt{m}}}}{\int_D e^{\frac{\langle \mathsf{Q}\mathsf{v}(z), \mathsf{K}\mathsf{v}(y) \rangle}{\sqrt{m}}} dz} \mathsf{V}\mathsf{v}(y) dy.$$

Attention as a Neural Operator: II

- Operator Attention $\mathbb{A} : C(D, \mathbb{R}^n) \mapsto C(D, \mathbb{R}^n)$ with

$$\mathrm{u}(x) = \mathbb{A}(\mathrm{v})(x) = \mathrm{W} \int_D \frac{e^{\frac{\langle \mathrm{Qv}(x), \mathrm{Kv}(y) \rangle}{\sqrt{m}}}}{\int_D e^{\frac{\langle \mathrm{Qv}(z), \mathrm{Kv}(y) \rangle}{\sqrt{m}}} dz} \mathrm{Vv}(y) dy.$$

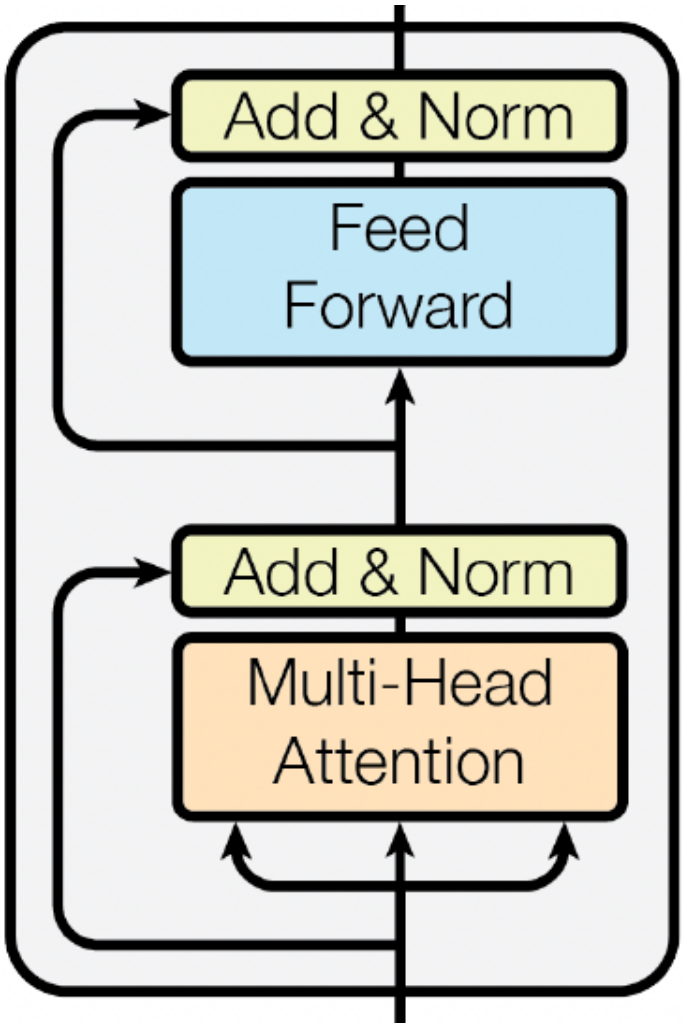
- Can be interpreted as as a Nonlinear Kernel Neural Operator:

$$\mathbb{A}(\mathrm{v})(x) = \int_D K(v(x), v(y)) v(y) dy.$$

- In contrast, FNO/CNO etc are Linear Kernel Neural Operators:

$$\mathbb{C}(\mathrm{v})(x) = \int_D K(x, y) v(y) dy = \int_D K(x - y) v(y) dy.$$

Additional Ingredients



Normalization ?

- Layer Normalization
- LayerNorm as a function:

$$LN : \mathbb{R}^n \mapsto \mathbb{R}^n, \quad LN(\mathrm{v}) = \alpha \odot \frac{\mathrm{v} - \mu_{\mathrm{v}}}{\sigma_{\mathrm{v}}} + \beta,$$

$$\mu_{\mathrm{v}} = \frac{1}{n}\sum_{j=1}^{n} \mathrm{v}_j, \quad \sigma_{bv}^2 = \frac{1}{n}\sum_{j=1}^{n} (\mathrm{v}_j - \mu_{bv})^2$$

- Straightforward to realize LayerNorm as a pointwise operator

$$(LN\mathrm{v})(x) = LN(\mathrm{v}(x))$$

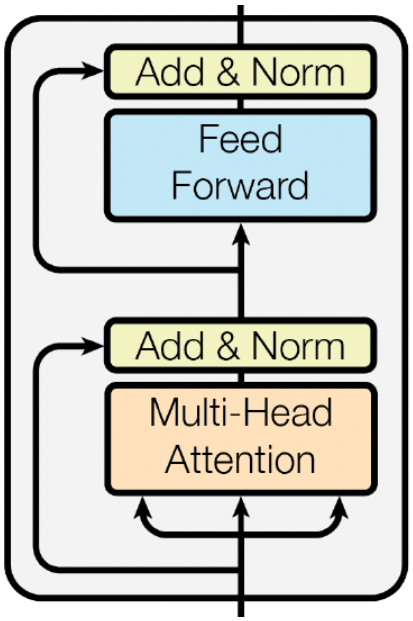
Feed Forward ?

- Use MLPs pointwise
- Operator interpretation is:

$$MLP(\mathrm{v})(x) = \overline{W}\sigma\left(W\mathrm{v}(x) + B\right), \quad \mathrm{v} \in \mathbb{R}^n$$

- with Weight Matrices $W \in \mathbb{R}^{d\times n}$, $\overline{W} \in \mathbb{R}^{n\times d}$
- and Bias vector $B \in \mathbb{R}^d$

An Operator Transformer Block



- A sequence of Operators of the form:
- Multi-head Self-Attention: $\bar{u} = MSA(v)$
- Residual + LayerNorm: $\hat{u} = LN(v + \bar{u}, \alpha_1, \beta_1)$
- MLP: $u' = MLP(\hat{u})$
- Residual + LayerNorm: $u = LN(u' + \hat{u}, \alpha_2, \beta_2)$

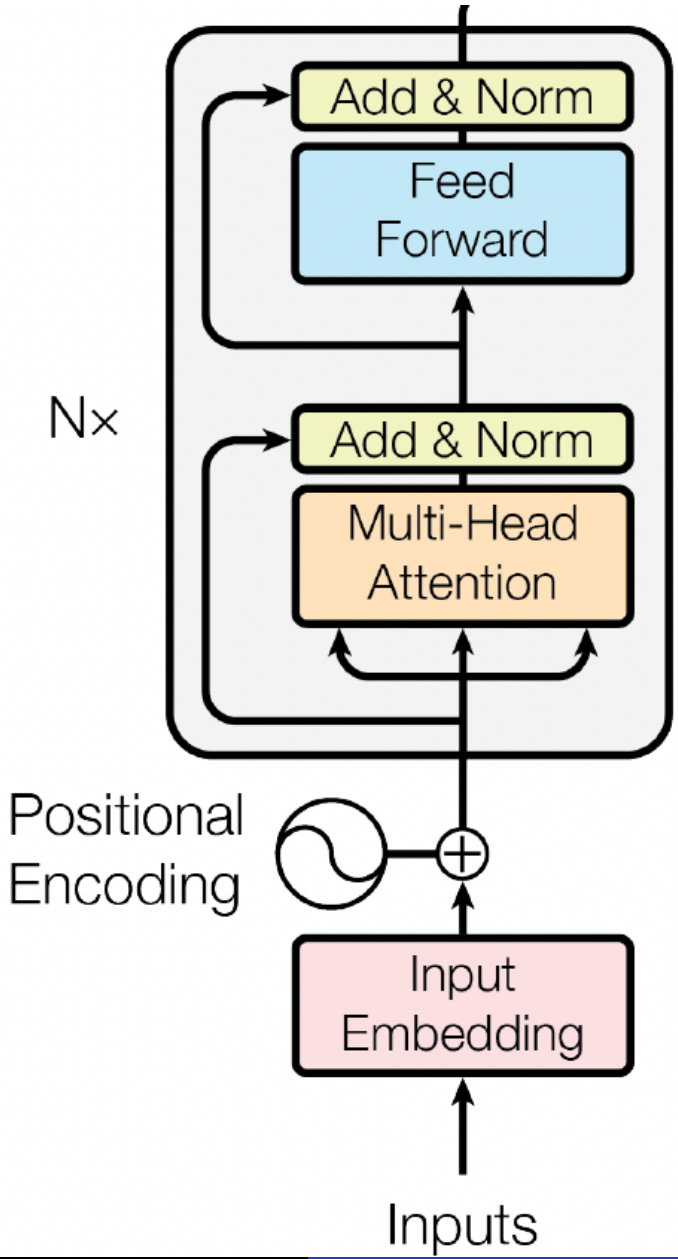
Issue: Where is Position ?

- All operators have Permutation Invariance.
- No sense of absolute or relative Position
- Have to include Positional encodings of form:

$$\hat{\mathrm{v}}_k = \overline{W}_v \sigma \left(W_v v_k \right) + \overline{W}_e \sigma \left(W_e e_k \right),$$

- With $e_k \in \mathbb{R}^K$ is the k-th unit vector
- Weight matrices
  $W_e \in \mathbb{R}^{d \times K}, \overline{W}_e \in \mathbb{R}^{n \times d}, W_v \in \mathbb{R}^{\bar{d} \times n}, \overline{W}_v \in \mathbb{R}^{n \times \bar{d}}$
- Fixed Sine-Cosine position encodings also work.

Final version of a Transformer Block



Caveat: Computational Complexity

- Computational Cost is Quadratic in # (Tokens) !!

$$\text{Compute} \sim \mathcal{O}(mnK^2)$$

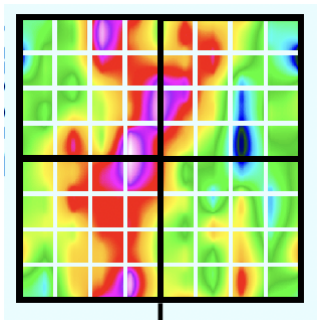
- With K- Input Length, n Input features and m hidden dimension.
- But lots of possible Parallelism in Computation
- $\mathcal{O}(1)$ sequential operations.
- $\mathcal{O}(1)$ Path Length.
- Nevertheless, Infeasible for 2 or 3-d inputs.

Possible Solution

- Vision Transformers (ViT) of Dosovitskiy et. al.
- Key Ingredient: Patching (Patchification):
- Given Image on resolution $H \times W$, i.e. $\mathrm{v} \in \mathbb{R}^{H \times H \times C}$
- Divide into $N = \frac{H^2}{p^2}$ patches, each of size $p \times p$
- $\mathrm{v} \sim [\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_N]$ with $\mathrm{v}_k \in \mathbb{R}^{p^2 \times C}$, $1 \leq k \leq N$
- Introduce Patch Embeddings $\mathrm{E} \in \mathbb{R}^{n \times (p^2 \cdot C)}$
- Input is a Sequence of Patch Tokens:

$$\hat{\mathrm{v}} = [\mathrm{E}\mathrm{v}_1, \mathrm{E}\mathrm{v}_2, \ldots, \mathrm{E}\mathrm{v}_N]$$

- N Tokens, each with n features are fed into a transformer !!



Operator Version

- Patch Formation + Embeddings can be written as operator:

$$\hat{\mathrm{E}}(\mathrm{v})(x) = \sum_{k=1}^{N} \mathrm{F} \left(\int_{D_k} W(x)\mathrm{v}(x)dx \right) \mathbb{I}_{D_k}(x).$$

- With $D = \cup_{k=1}^{N} D_k$ non-overlapping and equal partition.
- With learnable $F \in \mathbb{R}^{n \times C}$
- Weight Function $W(x) = \sum_{1 \le i,j \le H} W_{ij}\delta_{z_{ij}}$
- With uniform grid points z_{ij} and discrete weights defined by,

$$\begin{aligned} W_{ij} &= \omega_{ij} \quad \text{if} \quad 1 \le i,j \le p \\ &= \omega_{i \bmod p, j \bmod p}, \quad \text{otherwise.} \end{aligned}$$

ViT operator Block

- For $D \subset \mathbb{R}^2$ + input $\mathsf{v} \in C(D, \mathbb{R}^C)$
- A sequence of Operators of the form:
- Patch Embeddings+ Positional Encoding: $\hat{\mathsf{v}} = \hat{\mathsf{E}}(\mathsf{v}) + \mathsf{E}_{pos}(\mathsf{v})$
- LayerNorm + MSA+ Residual: $\bar{\mathsf{u}} = \hat{\mathsf{v}} + MSA(LN(\mathsf{v}))$
- LayerNorm + MLP+ Residual: $\mathsf{u} = \bar{\mathsf{u}} + MLP(LN(\bar{\mathsf{u}}))$

Computational Complexity

- Given an Image at resolution $H \times W$
- Standard Transformer needs

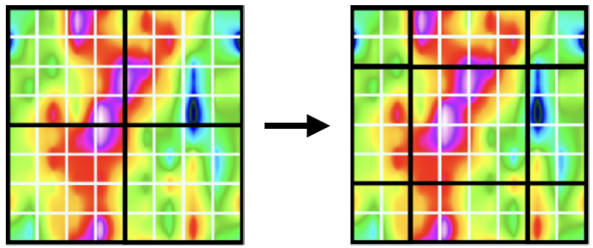
$$\text{Compute} \sim \mathcal{O}((HW)^2)$$

- ViT needs

$$\text{Compute} \sim \mathcal{O}\left(\frac{(HW)^2}{p^4}\right)$$

- Still not scalable for small patch size p

Another Idea: Windowed Attention of Liu et. al., 2022

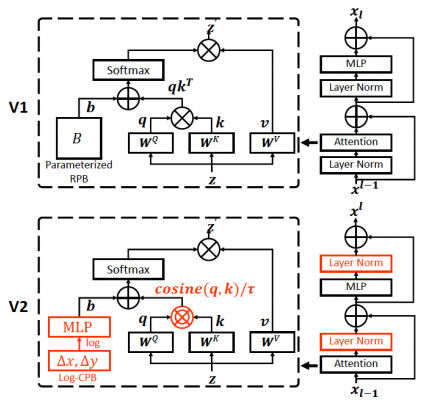


- Windowed Attention:

$$\mathbb{A}_W(\mathrm{v})(x) = \mathrm{W} \int\limits_{D_{q_x}^{w,\ell}} \frac{e^{\frac{\langle \mathrm{Qv}(x), \mathrm{Kv}(y) \rangle}{\sqrt{m}}}}{\int\limits_{D_{q_x}^{w,\ell}} e^{\frac{\langle \mathrm{Qv}(z), \mathrm{Kv}(y) \rangle}{\sqrt{m}}} dz} \mathrm{Vv}(y) dy.$$

- With M-Windows, $\mathrm{Compute} \sim \mathcal{O}\left(\frac{HWM^2}{p^2}\right)$
- How to Attend to Patches outside the Window ?
- Solution: Window shifts across Layers !!

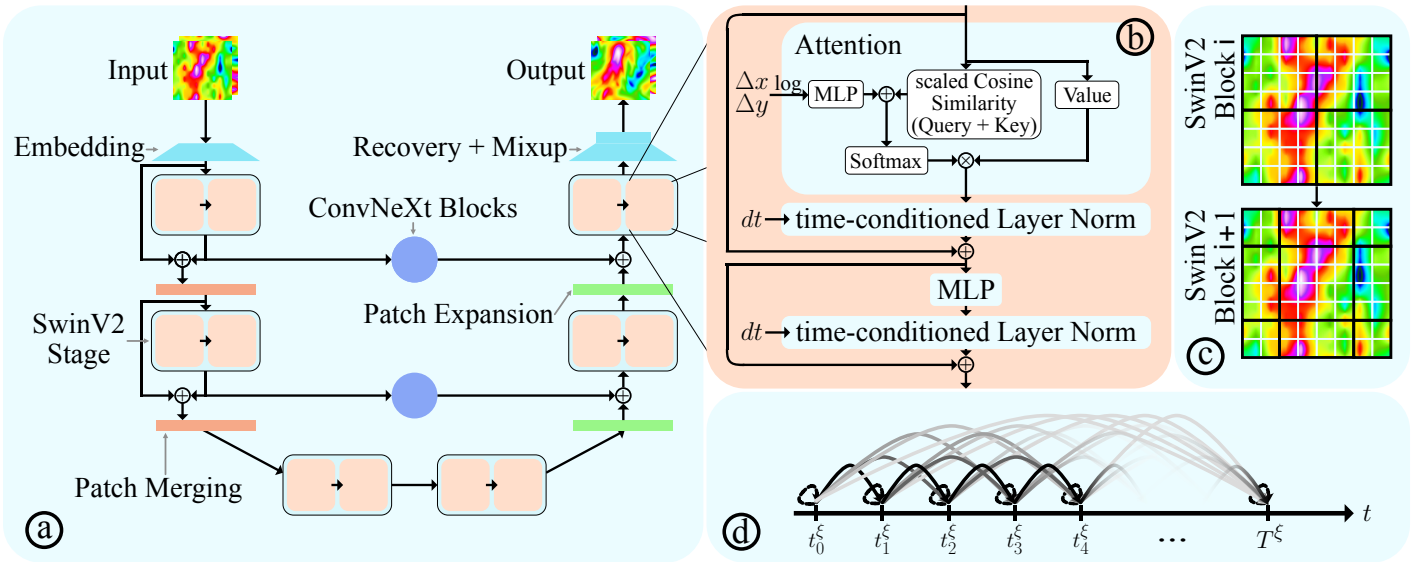
Modifications for Scalability



- Replace scaled dot product with Scaled Cosine
- Use MLPs on Relative Position Coordinates to generate positional encodings.

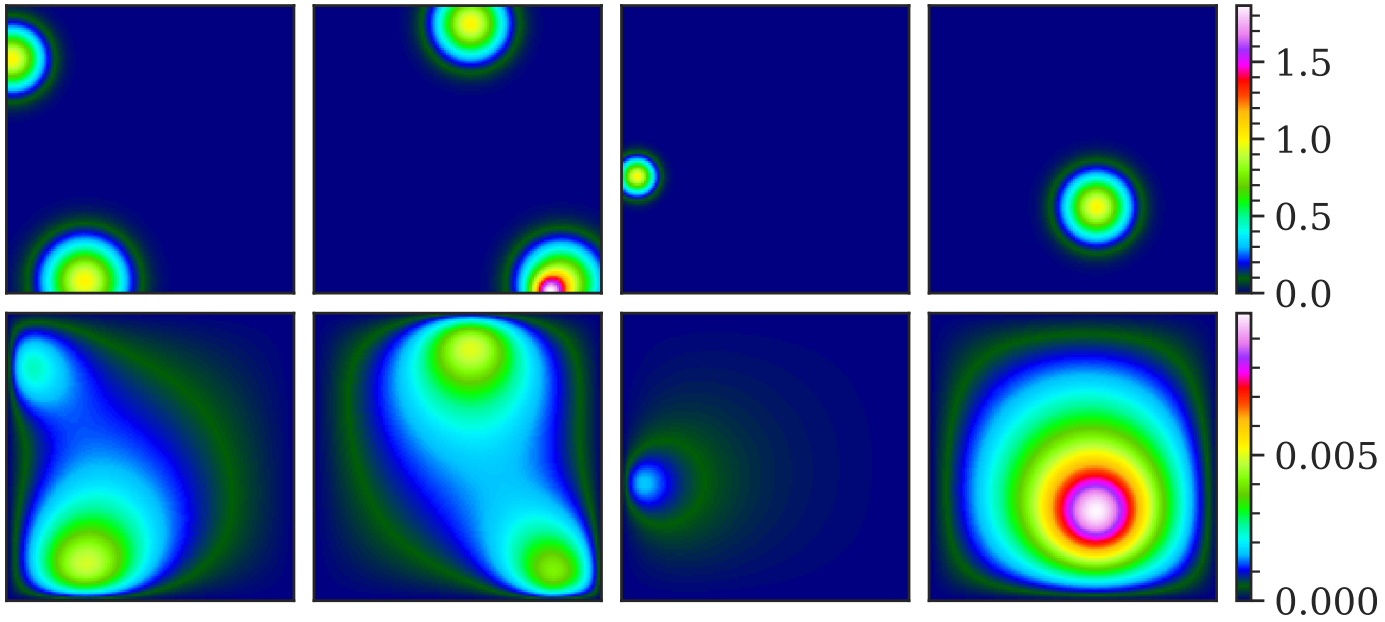
scOT: scalable Operator Transformer

- Proposed in Herde,SM et. al., 2024.

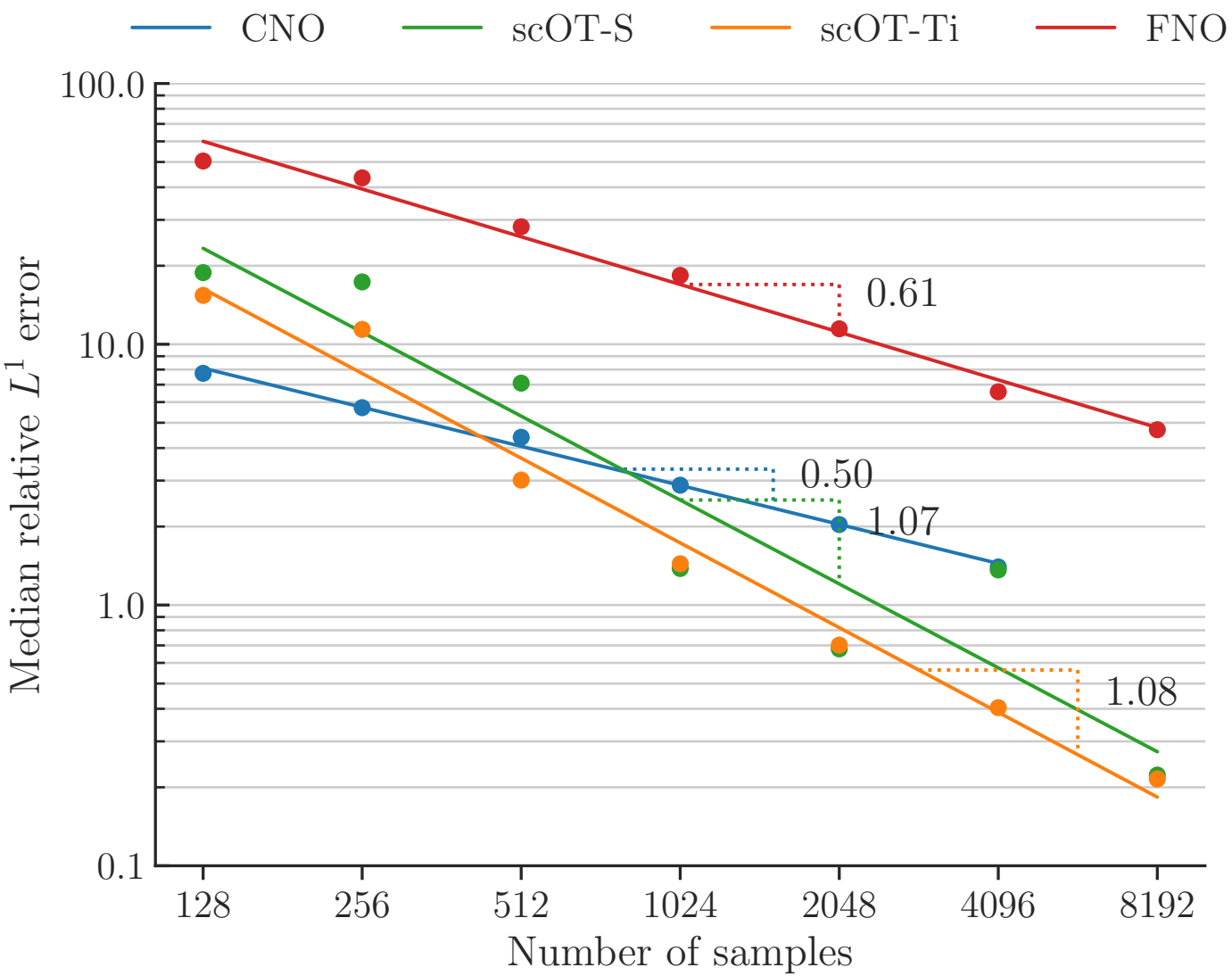


- scalable Operator Transformer: Based on SWIN Attention.
- Universal Approximator of Continuous Operators

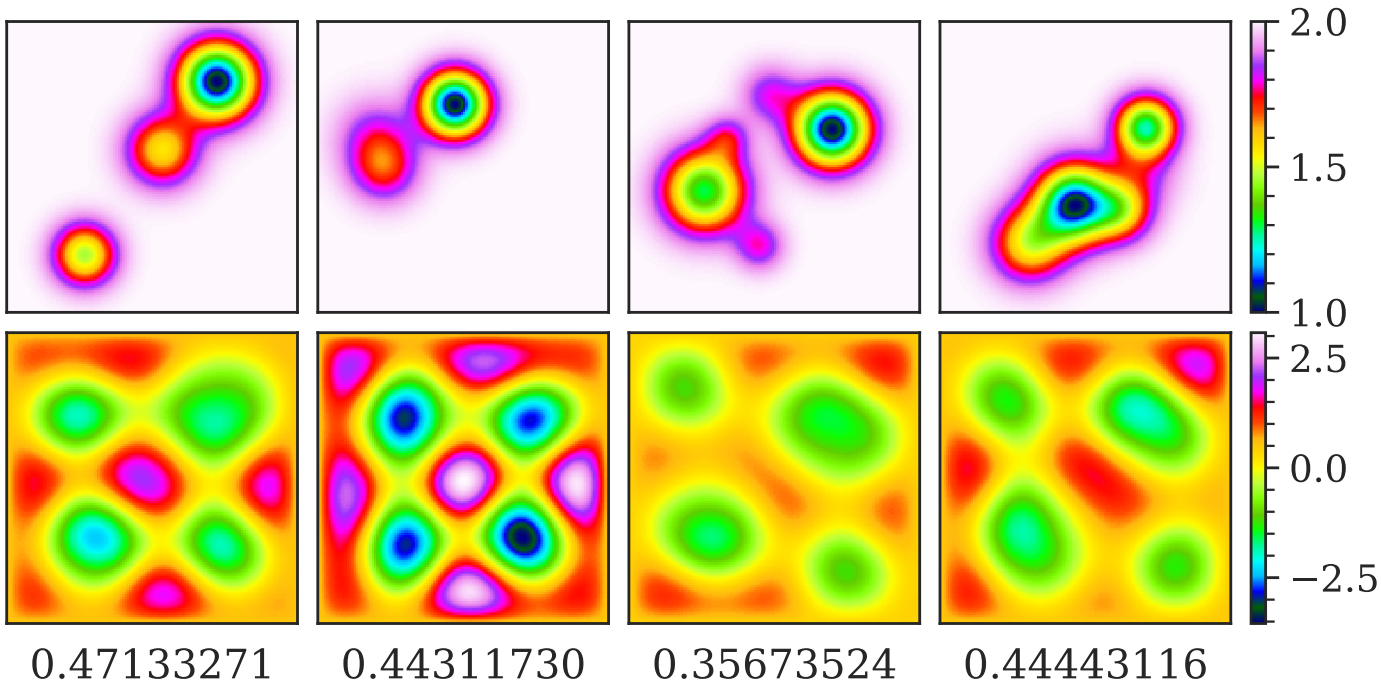
Poisson with Gaussian Sources



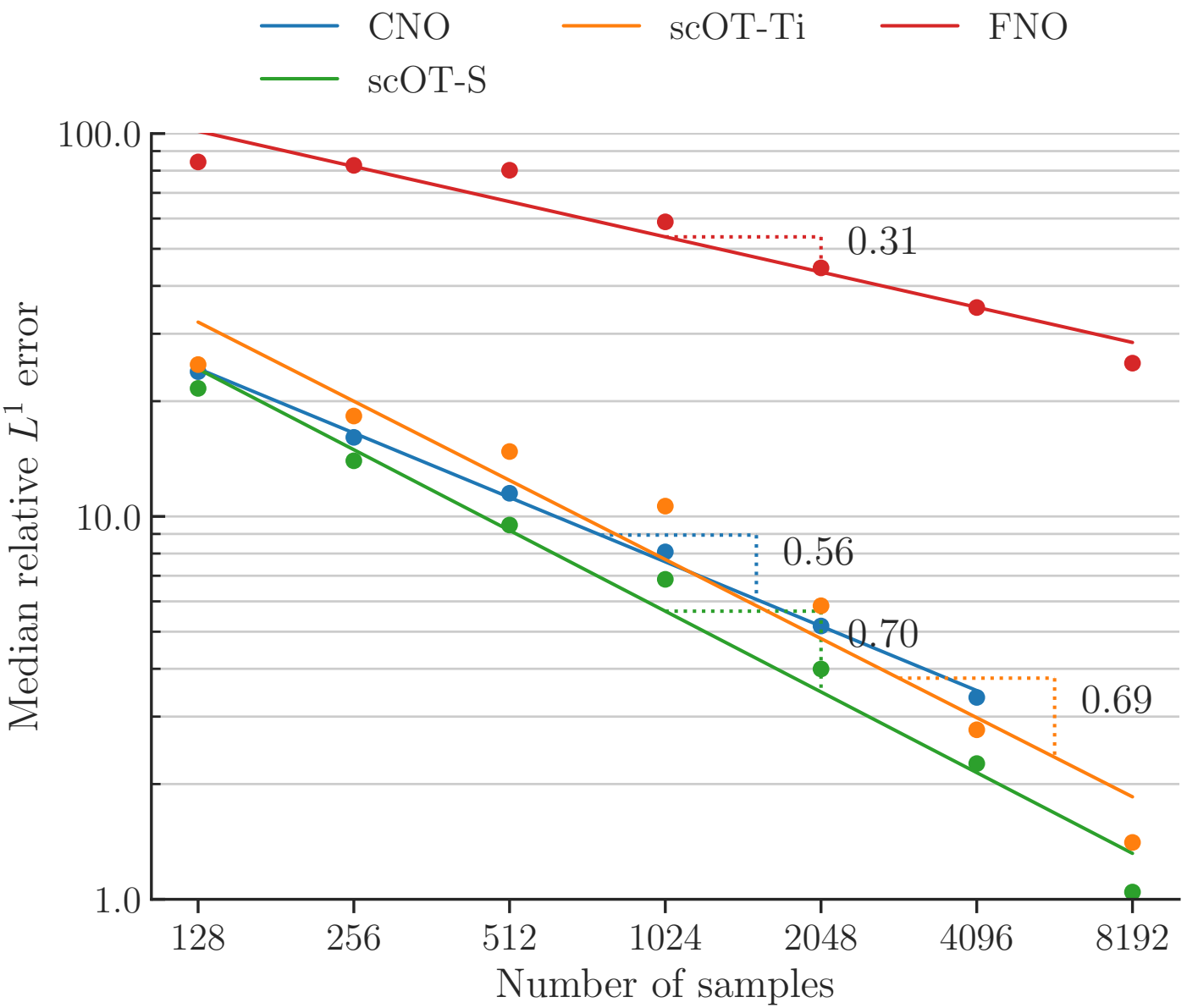
Poisson with Gaussian Sources



Helmholtz

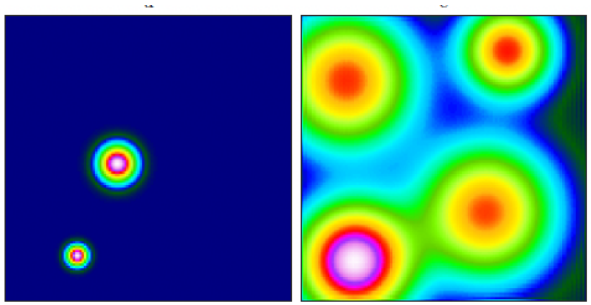


Helmholtz

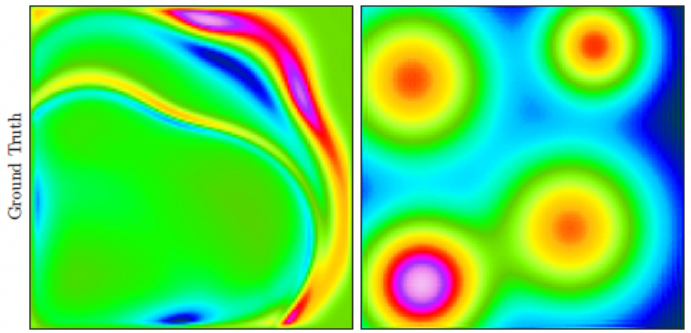

CNO
scOT-Ti
FNO
scOT-S
100.0
10.0
1.0
Median relative L^1 error
0.31
0.56
0.70
0.69
128
256
512
1024
2048
4096
8192
Number of samples

Wave Equation

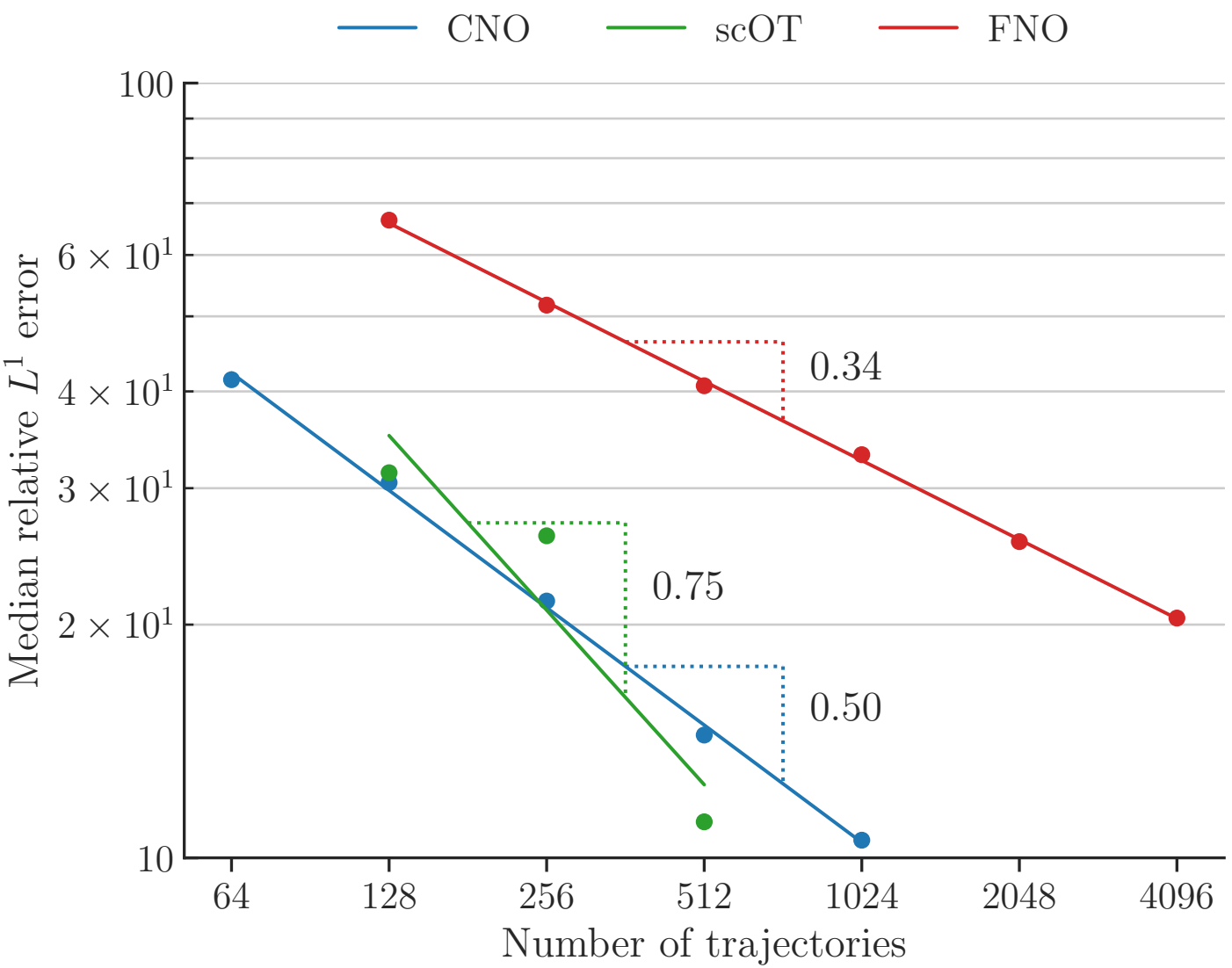
- Input ($T = 0$):



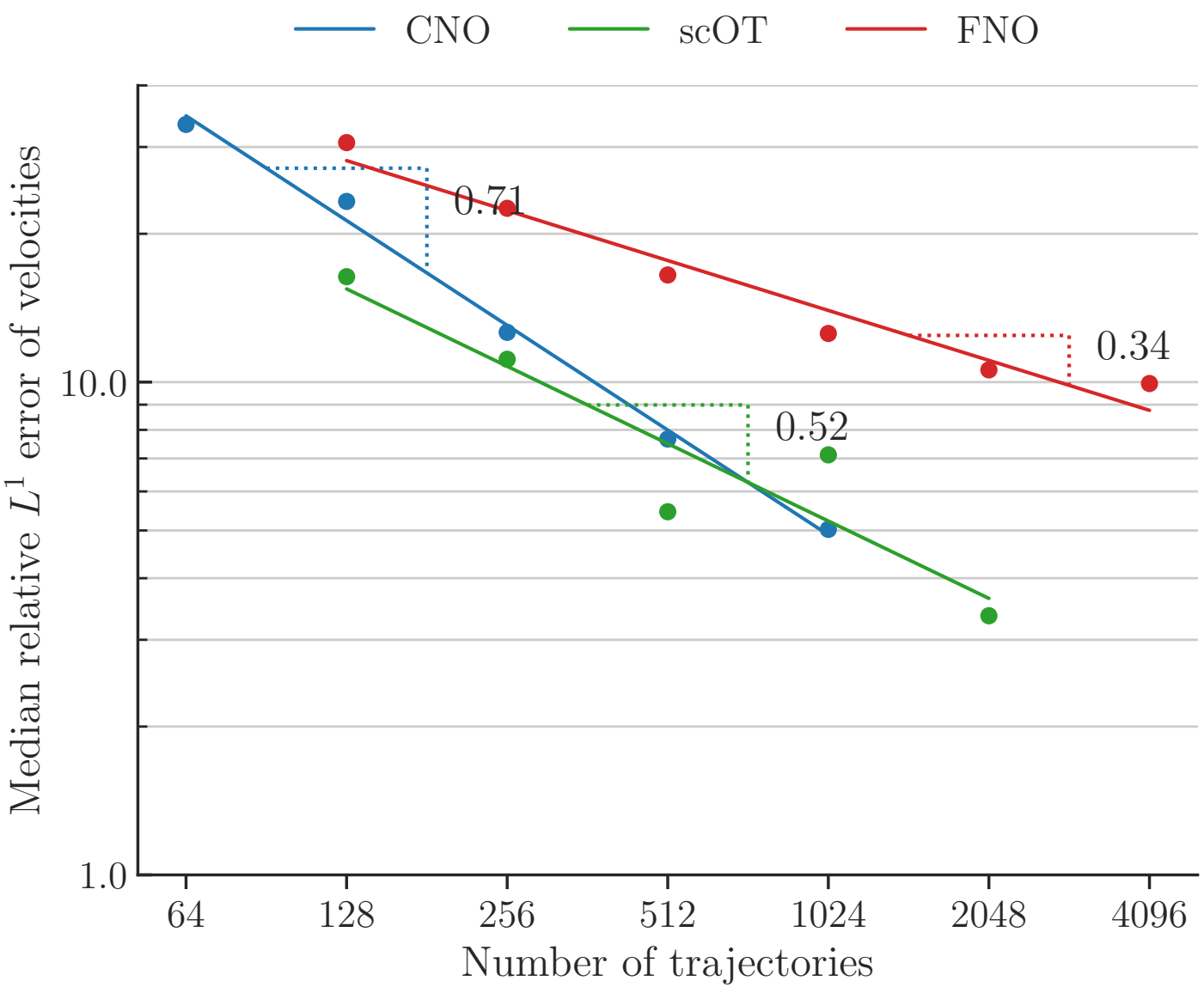
- Output t ($T = 1$):



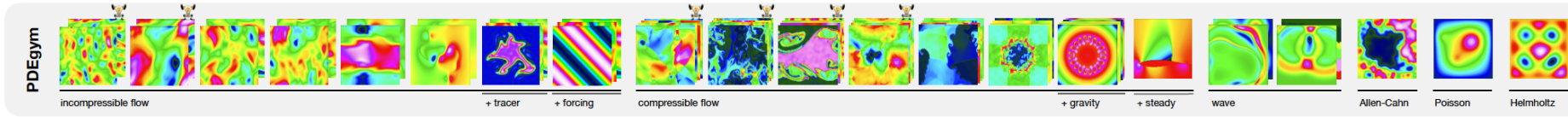
Wave Equation



Navier-Stokes: Shear Layer



CNO vs. scOT vs. FNO



- Performance on PDEgym Dataset.
- Set of 15 PDE operator learning tasks.

Model	Median (EG)	Mean (AG)	N(EG)	N(AG)
FNO	1	1	-	-
CNO	4.6	2.61	5	6
scOT	5.4	2.56	4	8

- Sample Complexity remains a problem !!