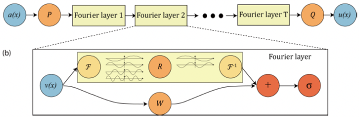
AI in the Sciences and Engineering HS 2025: Lecture 7

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.
- Discretize then Learn (Sample-CNN-Interpolate) did not generalize across grid resolutions.
- Learn then Discretize with Neural Operators.
- FNO is a prominent example.



- FNO also does not generalize across grid resolutions.
- Analyzed through the prism of RENOs

ReNO

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{E}"] & \mathcal{Y} \arrow[d, bend left, "\mathcal{E}"] \\
X \arrow[r, "G"] \arrow[u, bend left, "\mathcal{R}"] & Y \arrow[u, bend left, "\mathcal{R}"]
\end{tikzcd}$$

$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E}$$

- Discretize, then Learn $\Leftrightarrow$ Learn, then Discretize
- Following Bartolucci et al SM, 2023
- Aliasing error: $\varepsilon(\mathcal{G}, G) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E}$
- Representation Equivalent Neural Operator alias ReNO:

$$\varepsilon(\mathcal{G}, G) \equiv 0.$$

- Concept is instantiated Layerwise: $\mathcal{G} = \mathcal{G}_L \circ \cdots \mathcal{G}_\ell \cdots \mathcal{G}_1$:

$$\mathcal{G}_\ell - \mathcal{R} \circ G_\ell \circ \mathcal{E}$$

A ReNO on different Grids

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{E}"] & \mathcal{Y} \arrow[d, bend left, "\mathcal{E}"] \\
X \arrow[r, "G"] \arrow[u, bend left, "\mathcal{R}"] & Y \arrow[u, bend left, "\mathcal{R}"]
\end{tikzcd}$$

$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E}$$

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{E}'"] & \mathcal{Y} \arrow[d, bend left, "\mathcal{E}'"] \\
X' \arrow[r, "G'"] \arrow[u, bend left, "\mathcal{R}'"] & Y' \arrow[u, bend left, "\mathcal{R}'"]
\end{tikzcd}$$

$$G' = \mathcal{E}' \circ \mathcal{G} \circ \mathcal{R}'$$

- A Natural change of representation (Grid) Formula:
- As $\varepsilon(\mathcal{G}, G) \equiv 0 \equiv \varepsilon(\mathcal{G}, G')$.
- Aliasing $\Rightarrow$ Discrepancies between Resolutions !!

$$\begin{tikzcd}
X \arrow[r, "G"] & Y \arrow[d, bend left, "\mathcal{E} \circ \mathcal{R}'"] \\
X' \arrow[r, "G'"] \arrow[u, bend left, "\mathcal{E}' \circ \mathcal{R}"] & Y'
\end{tikzcd}$$

$$G' = \mathcal{E}' \circ \mathcal{R} \circ G \circ \mathcal{E} \circ \mathcal{R}'$$

A Concrete Example: 1-D on a Regular Grid

$$
\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend left, "\mathcal{E}"] & \mathcal{Y} \arrow[d, bend left, "\mathcal{E}"] \\
X \arrow[r, "G"] \arrow[u, bend left, "\mathcal{R}"] & Y \arrow[u, bend left, "\mathcal{R}"]
\end{tikzcd}
$$

$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E}$$

- $\mathcal{X}, \mathcal{Y}$ are Bandlimited Functions: i.e., $\mathrm{supp}\ \hat{u} \subset [-\Omega, \Omega]$
- Encoding is Pointwise evaluation: $\mathcal{E}(u) = \{u(x_j)\}_{j=1}^{n}$
- Reconstruction in terms of sinc basis:

$$\mathcal{R}(v)(x) = \sum_{j=1}^{n} v_j \mathrm{sinc}\,(x - x_j)$$

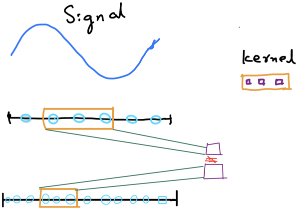
- Nyquist-Shannon $\Rightarrow$ bijection between $\mathcal{X}, X$ on sufficiently dense grid.
- Classical Aliasing Error: $\varepsilon(\mathcal{G}, G) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E}$

CNNs are not ReNOs !

- CNNs rely on Discrete Convolutions with fixed Kernel:

$$K_c[m] = \sum_{i=-s}^{s} k_i c[m-i]$$

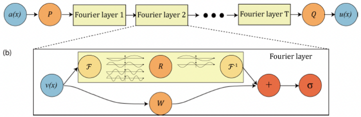
- Pointwise evaluations with Sinc basis



- Easy to check that CNNs are Resolution dependent as:

$$\mathcal{G}' \neq \mathcal{E}' \circ \mathcal{R} \circ \mathcal{G} \circ \mathcal{E} \circ \mathcal{R}'$$

Are FNOs ReNOs ?

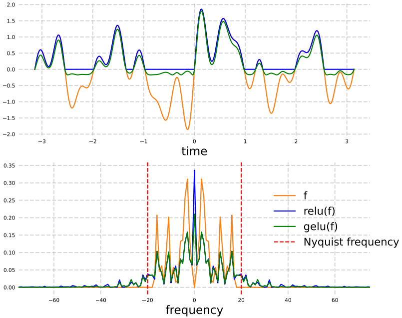


- Convolution in Fourier space $\mathcal{K}$ + Nonlinearity σ
- $\mathcal{K}$ is ReNO wrt Periodic Bandlimited functions $\mathcal{P}_K$:

$$\begin{tikzcd}
\mathcal{P}_K \arrow[r, "\mathcal{F}"] \arrow[d, "T^{\dagger}_{\Psi_K}"] & \mathbb{C}^{2K+1} \arrow[r, "R\odot"] \arrow[d, leftrightarrow, "\mathrm{Id}"'] & \mathbb{C}^{2K'+1} \arrow[r, "\mathcal{F}^{-1}"] \arrow[d, leftrightarrow, "\mathrm{Id}"'] & \mathcal{P}_{K'} \\
\mathbb{C}^{2K+1} \arrow[r, "(2K+1)\cdot\mathrm{F}"] & \mathbb{C}^{2K+1} \arrow[r, "R\odot"] & \mathbb{C}^{2K'+1} \arrow[r, "\frac{1}{(2K'+1)}\cdot\mathrm{F}^{-1}"] & \mathbb{C}^{2K'+1} \arrow[u, "T_{\Psi_{K'}}"]
\end{tikzcd}$$

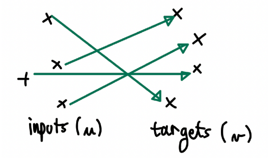
What about activations ?

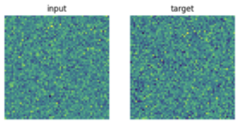
- Nonlinear activation σ can break bandlimits: $\sigma(f) \notin \mathcal{P}_K$
- FNOs are not necessarily ReNOs !!



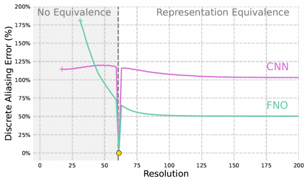
A Synthetic Example: Random Assignment

- The underlying Operator:

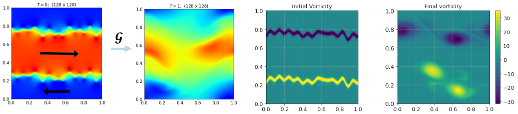




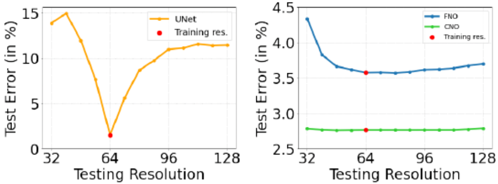
- Errors:



A Practical Example



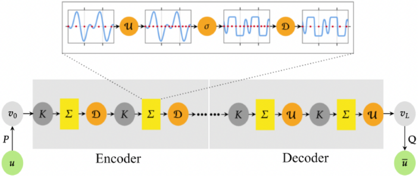
- FNO Results:



- Challenge: Design a ReNO

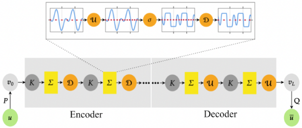
Convolutions Strike Back !!

- Convolutional Neural Operators (CNOs) of Raonic, SM et al, 2023.



- Operator between Band-Limited Functions
- Building Blocks:
- Lifting operator: P
- Projection operator: Q

CNO Key Building Block I



- Use Continuous Convolutions on Bandlimited functions
- Convolution Kernel is still Discrete !!
- Convolution operator is a ReNO.

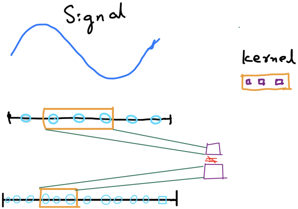
$$\begin{tikzcd} \mathcal{B}_w \arrow[r, "\mathcal{K}_w"] & \mathcal{B}_w \arrow[d, "T^*_{\Psi_w}"] \\ \ell^2(\mathbb{Z}^2) \arrow[u, "T_{\Psi_w}"] \arrow[r] & \ell^2(\mathbb{Z}^2) \end{tikzcd}$$

Contrast with CNNs

- CNNs rely on Discrete Convolutions with fixed Kernel:

$$K_c[m] = \sum_{i=-s}^{s} k_i c[m-i]$$

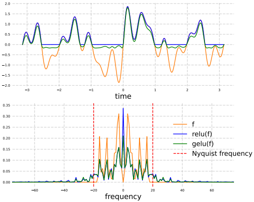
- Pointwise evaluations with Sinc basis

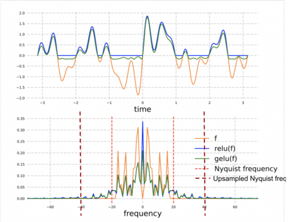


- Easy to check that CNNs are Resolution dependent as:

$$\mathcal{G}' \neq \mathcal{E}' \circ \mathcal{R} \circ \mathcal{G} \circ \mathcal{E} \circ \mathcal{R}'$$

CNO Key Building Block II: Activation Function ?

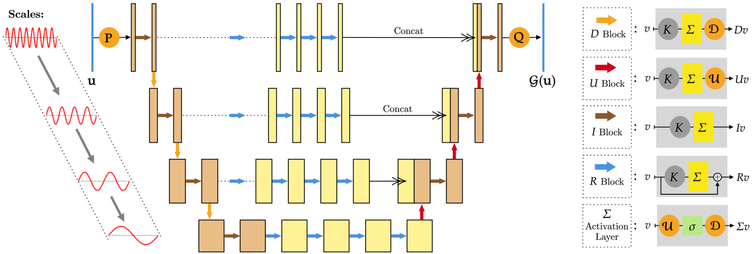




- Apply Activation as $\Sigma : B_w \mapsto B_w$ with $\Sigma = \mathcal{D}_{\bar{w},w} \circ \sigma \circ \mathcal{U}_{w,\bar{w}}$
- Upsampling: $\mathcal{U}_{w,\bar{w}} f = f$ with $w < \bar{w}$
- Downsampling: $\mathcal{D}_{\bar{w},w} f(x) = \left(\frac{\bar{w}}{w}\right)^d \int\limits_D \mathrm{sinc}(2\bar{w}(x-y)) f(y) dy$
- Activation is a ReNO if $\bar{w} >> w$:

$$\begin{tikzcd}
\mathcal{B}_w \arrow[r, hook, "P_{\mathcal{B}_{\overline{w}}(\mathbb{R}^2)}"] & \mathcal{B}_{\overline{w}} \arrow[r, "\sigma"] \arrow[d, "T^*_{\Psi_{\overline{w}}}"] & \mathcal{B}_{\overline{w}} \arrow[r, "P_{\mathcal{B}_w(\mathbb{R}^2)}"] & \mathcal{B}_w \arrow[d, "T^*_{\Psi_w}"] \\
\ell^2(\mathbb{Z}^2) \arrow[u, "T_{\Psi_w}"] \arrow[r, "\mathcal{U}_{w,\overline{w}}"] & \ell^2(\mathbb{Z}^2) \arrow[r, "\sigma"] & \ell^2(\mathbb{Z}^2) \arrow[u, "T_{\Psi_{\overline{w}}}"] \arrow[r, "\mathcal{D}_{\overline{w},w}"] & \ell^2(\mathbb{Z}^2)
\end{tikzcd}$$

CNO Architecture in Practice



- CNO instantiated as a modified Operator UNet
- Built for multiscale information processing

CNO properties

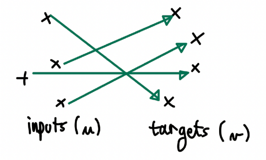
- CNO is a ReNO by construction.
- Universal Approximation Theorem:
- CNOs approximate any Continuous + operators $\mathcal{G} : H^r \mapsto H^s$
- Proof relies on building $\mathcal{G} \approx \mathcal{G}^* : \mathcal{B}_w \mapsto \mathcal{B}_{w'}$

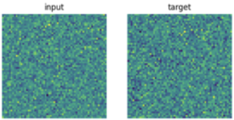
$$\begin{tikzcd}
\mathcal{B}_w \arrow[r, "\mathcal{G}^*"] \arrow[d, "T^*_{\Psi_w}"] & \mathcal{B}_{w'}, \\
\ell^2(\mathbb{Z}^2) \arrow[r, "\mathfrak{G}_{\Psi_w, \Psi_{w'}}"] & \ell^2(\mathbb{Z}^2) \arrow[u, "T_{\Psi_{w'}}"']
\end{tikzcd}$$

- Efficient PyTorch implementation with CUDA kernels.
- Code available on https://github.com/bogdanraonic3/ConvolutionalNeuralOperator.git

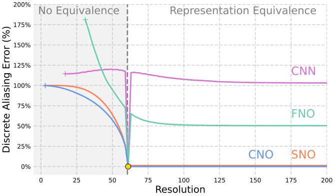
A Synthetic Example: Random Assignment

- The underlying Operator:



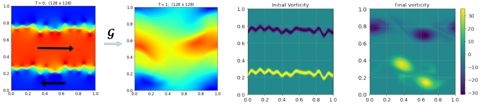


- Errors:

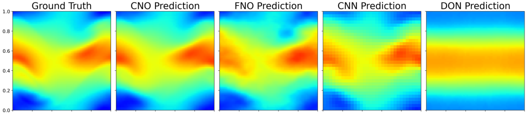


Ex 1: Navier-Stokes Eqns.

- Operator:



- Comparison:

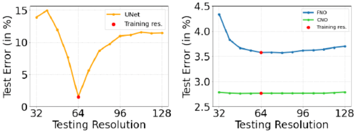


- Test Errors:

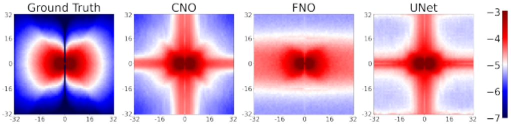
Model	FFNN	UNet	DeepONet	FNO	CNO
Error	8.05%	3.54%	11.64%	3.93%	3.01%

Further Results

- Resolution Dependence:

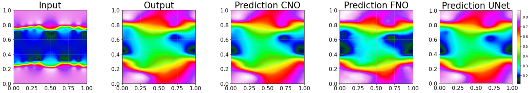


- Spectral Behavior: log spectra

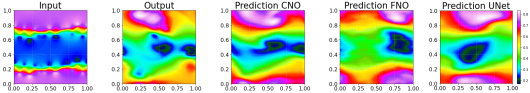


Out-of-Distribution Generalization or Zero-shot Learning

- Results for In-Distribution Testing:



- Results for Out-of-Distribution Testing:

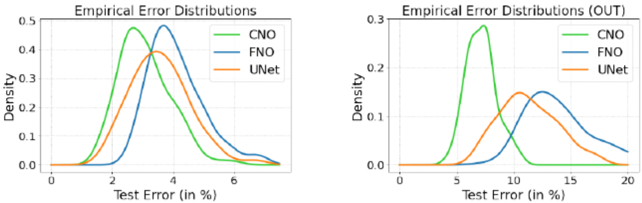


- Test Errors:

Model	FFNN	UNet	DeepONet	FNO	CNO
In	8.05%	3.54%	11.64%	3.93%	3.01%
Out	16.12%	10.93%	15.05%	13.45%	7.06%

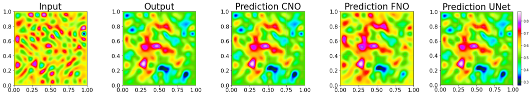
- RunTime: 10^{-1}s on 100^2 grid for AzeBan vs 10^{-4}s for CNO

Success is a histogram, not a point !!

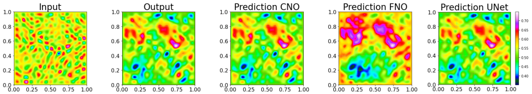


Ex II: Poisson Eqn

- PDE: $-\Delta u = f$, Operator: $\mathcal{S} : f \mapsto u$.
- Data: $f \sim \sum_{i,j=1}^{K} \frac{a_{ij}}{(i^2+j^2)^{\alpha}} \sin(i\pi x)\sin(j\pi y), \;\; a_{ij} \sim \mathcal{U}[-1,1]$
- Results for In-Distribution Testing: $K = 16$



- Results for Out-of-Distribution Testing: $K = 20$

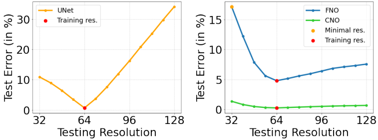


- Test Errors:

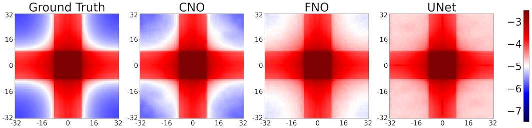
Model	FFNN	UNet	DeepONet	FNO	CNO
In	5.74%	0.71%	12.92%	4.78%	0.23%
Out	5.35%	1.27%	9.15%	8.89%	0.27%

Further Results

- Resolution Dependence:

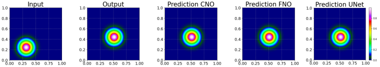


- Spectral Behavior: log spectra

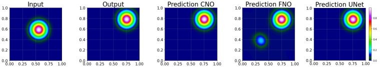


Ex 3: Transport

- Results for In-Distribution Testing:

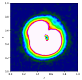


- Results for Out-of-Distribution Testing:

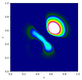


- Test Errors:

Model	FFNN	UNet	DeepONet	FNO	CNO
In	7.09%	0.49%	1.14%	0.40%	0.30%
Out	650.57%	1.28%	157.22%	13.83%	0.47%



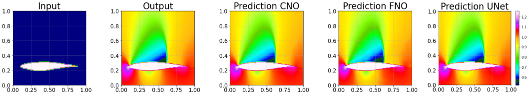
FFNN



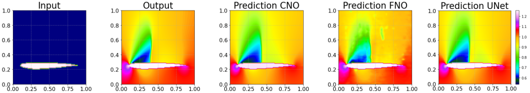
DeepONet

Ex 4: Compressible Euler Eqns

- Results for In-Distribution Testing:



- Results for Out-of-Distribution Testing:



- Test Errors:

Model	FFNN	UNet	DeepONet	FNO	CNO
In	0.78%	0.38%	1.93%	0.47%	0.35%
Out	1.34%	0.76%	2.88%	0.85%	0.62%

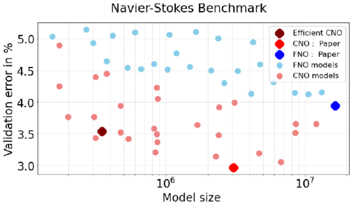
- RunTime: 10^2s for NuwTun vs 10^{-4}s for CNO

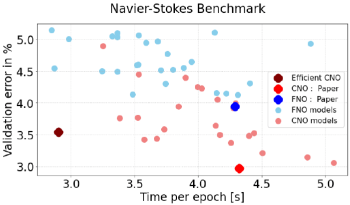
Similar Performance across the board !!

- Extensive Empirical evaluation on RPB benchmarks.

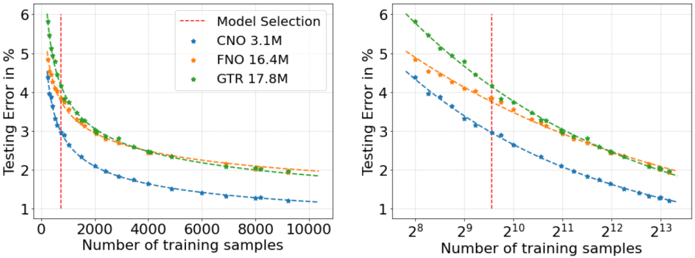
	In/Out	FFNN	GT	UNet	ResNet	DON	FNO	CNO
Poisson Equation	In	5.74%	2.77%	0.71%	0.43%	12.92%	4.98%	**0.21%**
	Out	5.35%	2.84%	1.27%	1.10%	9.15%	7.05%	**0.27%**
Wave Equation	In	2.51%	1,44%	1.51%	0.79%	2.26%	1.02%	**0.63%**
	Out	3.01%	1.79%	2.03%	1.36%	2.83%	1.77%	**1.17%**
Smooth Transport	In	7.09%	0.98%	0.49%	0.39%	1.14%	0.28%	**0.24%**
	Out	650.6%	875.4%	1.28%	0.96%	157.2%	3.90%	**0.46%**
Discontinuous Transport	In	13.0%	1.55%	1.31%	**1.01%**	5.78%	1.15%	**1.01%**
	Out	257.3%	22691.1%	1.35%	1.16%	117.1%	2.89%	**1.09%**
Allen-Cahn Equation	In	18.27%	0.77%	0.82%	1.40%	13.63%	**0.28%**	0.54%
	Out	46.93%	2.90%	2.18%	3.74%	19.86%	**1.10%**	2.23%
Navier-Stokes Equations	In	8.05%	4.14%	3.54%	3.69%	11.64%	3.57%	**2.76%**
	Out	16.12%	11.09%	10.93%	9.68%	15.05%	9.58%	**7.04%**
Darcy Flow	In	2.14%	0.86%	0.54%	0.42%	1.13%	0.80%	**0.38%**
	Out	2.23%	1.17%	0.64%	0.60%	1.61%	1.11%	**0.50%**
Compressible Euler	In	0.78%	2.09%	0.38%	1.70%	1.93%	0.44%	**0.35%**
	Out	1.34%	2.94%	0.76%	2.06%	2.88%	0.69%	**0.59%**

Computational Efficiency of CNO

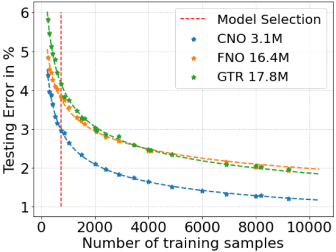


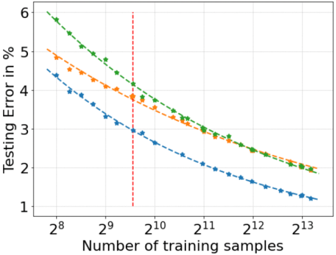


How does CNO/FNO Scale ?



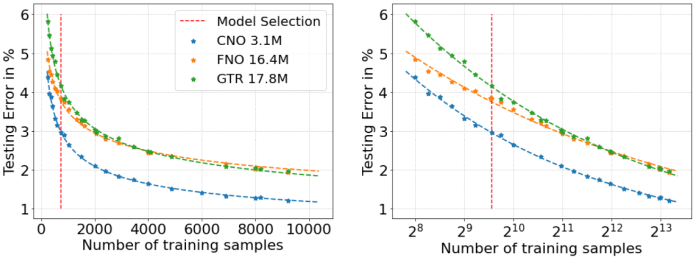
How does CNO/FNO Scale ?





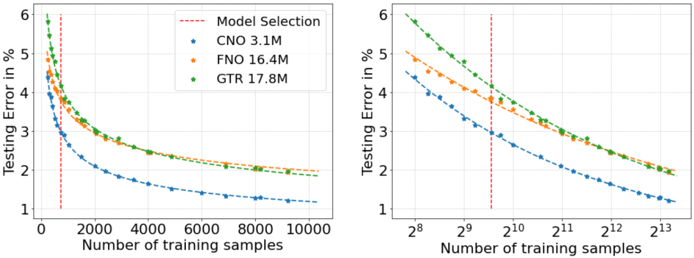
- Success is a curve, not a point !!

How does CNO/FNO Scale ?



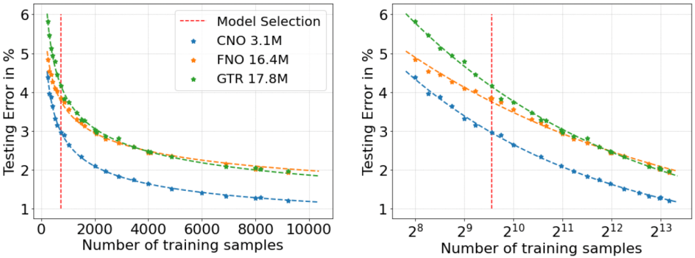
- Success is a curve, not a point !!
- Models Scale with sample size: $\mathcal{E} \sim N^{-\alpha}$ but with α small
- Theory: Lanthaler, SM, Karniadakis, De Ryck, SM,

How does CNO/FNO Scale ?



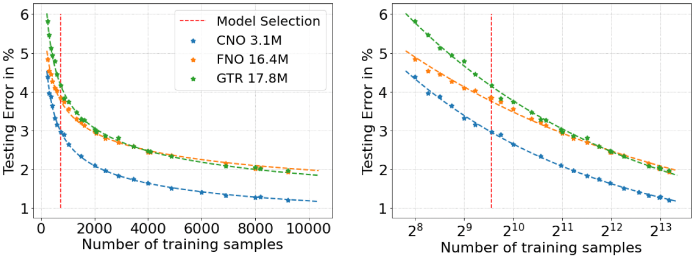
- Success is a curve, not a point !!
- Models Scale with sample size: $\mathcal{E} \sim N^{-\alpha}$ but with α small
- Theory: Lanthaler, SM, Karniadakis, De Ryck, SM,
- ML models require Big Data: $\mathcal{O}(10^3) - \mathcal{O}(10^4)$ training samples per Task
- Very Difficult to obtain Data for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

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 about Nonlinear Kernels ?