AI in the Sciences and Engineering HS 2025: Lecture 6

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.

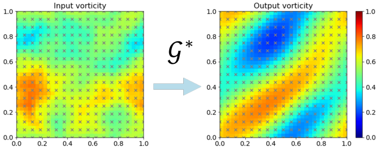
What we have learnt so far ?

- AIM: Learn/Solve PDEs using Deep Neural Networks
- PINNs have limitations.
- Use Data-driven Operator Learning instead.

What is Operator Learning ?

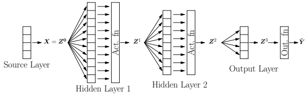
- Operator: $\mathcal{G} : \mathcal{X} \mapsto \mathcal{Y}$, $\dim\,(\mathcal{X}, \mathcal{Y}) = \infty$.
- Learn PDE Solution Operators from Data
- Underlying Data Distribution $\mu \in \mathrm{Prob}\,(\mathcal{X})$
- Draw N i.i.d samples $(a_i, \mathcal{G}(a_i))$ with $a_i \sim \mu$.
- Operator Learning Task: Find approximation to $\mathcal{G}_{\#}\mu$

Solution I: Discretize, then Learn !!



- Discretization $\mapsto$ MLP $\mapsto$ Interpolation
- Solution operator $\mathcal{G} : \mathcal{X} \mapsto \mathcal{X}$
- Discretization: $\mathcal{E} : \mathcal{X} \mapsto \mathcal{X}^{\Delta} \sim \mathbb{R}^N$
- MLP: $\mathcal{L}^* : \mathbb{R}^N \mapsto \mathbb{R}^N$
- Interpolation: $\mathcal{R} : \mathbb{R}^N \mapsto \mathcal{X}$
- Operator Learning: $\mathcal{G}^* = \mathcal{R} \circ \mathcal{L}^* \circ \mathcal{E}$

Deep Neural networks Recalled



- $\mathcal{L}^*(z) = \sigma_o \odot C_K \odot \sigma \odot C_{K-1} \ldots\ldots\ldots \odot \sigma \odot C_2 \odot \sigma \odot C_1(z)$.
- At the k-th Hidden layer: $z^{k+1} := \sigma(C_k z^k) = \sigma(W_k z^k + B_k)$
- Trainable Parameters: $\theta = \{W_k, B_k\} \in \Theta$,
- σ: scalar Activation function: ReLU, Tanh
- Random Training set: $\mathcal{S} = \{z_i\}_{i=1}^N \in Z \subset \mathbb{R}^N$, with i.i.d z_i
- Use GD: $\theta_{k+1} = \theta_k - \eta_k \nabla_\theta J(\theta_k)$, to find

$$\theta^* := \arg\min_{\theta \in \Theta} \underbrace{\sum_{i=1}^{N} |\mathcal{L}(z_i) - \mathcal{L}^*_\theta(z_i)|^p}_{J(\theta)},$$

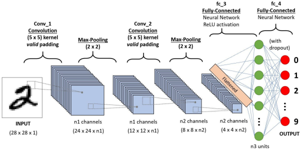
- with $\mathcal{G} = \mathcal{R} \circ \mathcal{L} \circ \mathcal{E}$

Issues

- Large Model Size:
  - For discretization on a 64^2 grid.
  - $N = 4096 \Rightarrow$ Single Hidden Layer of dimension $\mathcal{O}(10^8)$.
  - Too large models even for shallow MLPs.
- Spatial Structure is not incorporated !!
- Not possible to evaluate on a Different Resolution
- Not genuine operator learning.

Solution I (Refined)

- Use CNNs instead of MLP.

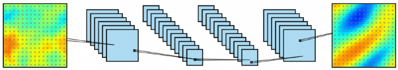


- Key elements are Discrete Convolutions with fixed Kernel:

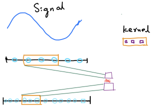
$$K_c[m] = \sum_{i=-s}^{s} k_i c[m-i]$$

- Weight matrices are very Sparse

Operator Learning with CNNs

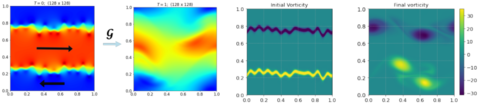


- Tractable Model sizes.
- Respect Spatial structure of underlying Data.
- Possible to Evaluate on any grid resolution



Does this work ?

- Shear flow with Navier-Stokes with $Re >> 1$



- CNN + Interpolation Results:

- Consistent with Zhu,Zabaras, 2019.
- Desiderata for Operator Learning:
- Input + Output are functions.
- Learn underlying Operator, not just a discrete Representation

Solution II: Learn, then Discretize !!

- Use Neural Operators
- Formalized in Kovachki et al, 2021.
- Recall: DNNs are $\mathcal{L}_\theta = \sigma_K \odot \sigma_{K-1} \odot \ldots\ldots \sigma_1$
- Single hidden layer: $\sigma_k(y) = \sigma(A_k y + B_k)$
- Neural Operators generalize DNNs to ∞-dimensions:
- NO: $\mathcal{N}_\theta = \mathcal{N}_L \odot \mathcal{N}_{L-1} \odot \ldots\ldots \mathcal{N}_1$
- Single hidden layer; $\mathcal{N}_\ell : \mathcal{X} \mapsto \mathcal{X}$
- Need to find Function Space versions of
  - Bias Vector
  - Weight Matrix
  - Activation function

Neural Operators (Contd..)

- Replace Bias vector by Bias function $B_\ell(x)$
- Replace Matrix-Vector multiply by Kernel Integral Operators:

$$A_\ell y \rightarrow \int_D K_\ell(x,y)v(y)dy$$

- Pointwise activations results in:

$$(\mathcal{N}_\ell v)(x) = \sigma\left(\int_D K_\ell(x,y)v(y)dy + B_\ell(x)\right)$$

- Learning Parameters in B_ℓ, K_ℓ

Discrete Realization

- Caveat: Computational Complexity
- Different Kernels $\Rightarrow$ Low-Rank NOs, Graph NOs, Multipole NOs, ...

Fourier Neural Operators

- FNO proposed in Li et al, 2020.
- Translation invariant Kernel $K(x, y) = K(x - y)$
- Kernel Integral Operator is $\int_D K(x, y)v(y)dy = K * v$
- Key Trick: Perform Convolution in Fourier space
- Fourier Transform: $\mathcal{F} : L^2(D, \mathbb{C}^n) \mapsto l^2(\mathbb{Z}^d, \mathbb{C}^n)$

$$(\mathcal{F}v_j)(k) = \int_D v_j(x)\Psi_k(x)dx, \quad \Psi_k(x) = Ce^{-2\pi i\langle k,x\rangle}$$

- Inverse Fourier Transform: $\mathcal{F}^{-1} : l^2(\mathbb{Z}^d, \mathbb{C}^n) \mapsto L^2(D, \mathbb{C}^n)$

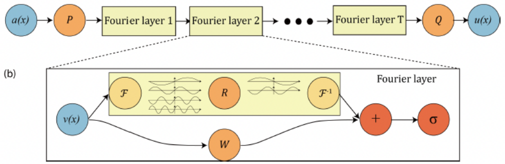
$$(\mathcal{F}^{-1}w_k)(x) = \sum_{k\in\mathbb{Z}^d} w_k\Psi_k(x)$$

FNO Details

- Use Fourier and Inverse Fourier Transform to define the KIO:

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

- Parametrize Kernel in Fourier space.
- With Fixed Number of Fourier Modes
- Fast implementation through FFT



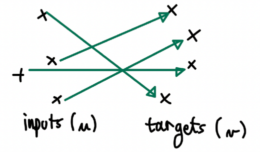
Theory for FNOs

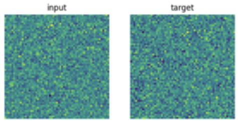
- FNO is very widely used in Practice.
- Theoretical Results of Kovachki, Lanthaler,SM, 2021
- Universal Approximation Thm: For $\mu \in Prob(L^2(D))$ and any measurable $\mathcal{G} : H^r \mapsto H^s$ and $\epsilon > 0$, $\exists \mathcal{N}$ (FNO): $\hat{\mathcal{E}} < \epsilon$
- With Error:

$$\hat{\mathcal{E}}^2 = \int\limits_X \int\limits_U |\mathcal{G}(u)(y)) - \mathcal{N}(u)(y)|^2 \, dy d\mu(u)$$

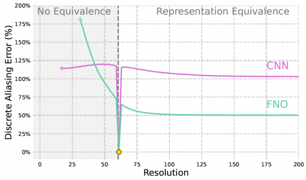
Is there a Catch ?

- A Synthetic Example: Random Assignment
- The underlying Operator:

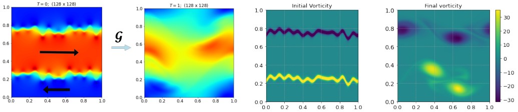




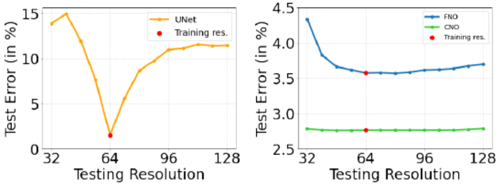
- Errors:



A Practical Example



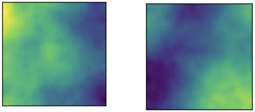
- FNO Results:



- Learn underlying Operator, not just a discrete Representation !!

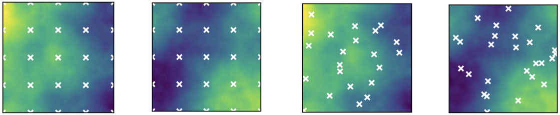
Why is this a challenge ?

- In principle, Operator maps functions to functions.



Input Output

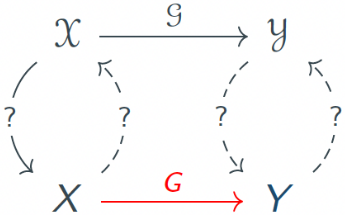
- In practice, both inputs and outputs are Discrete



Input Output Input Output

- Multiple Discrete Representations !!
- Only discrete operations on Digital Computers.
- A proper notion of Continuous-Discrete Equivalence (CDE)

On Continuous-Discrete Equivalence



Continuous-Discrete Equivalence (Contd...)

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

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

- 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

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

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

- 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:

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

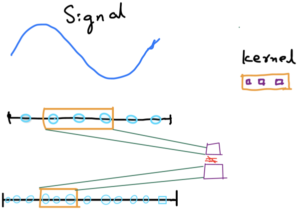
- Nyquist-Shannon $\Rightarrow$ bijection between $\mathcal{X}, X$ on sufficiently dense grid.
- Classical Aliasing Error: $\varepsilon(\mathcal{G}, G) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E}$

CNNs are not ReNOs !

- CNNs rely on Discrete Convolutions with fixed Kernel:

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

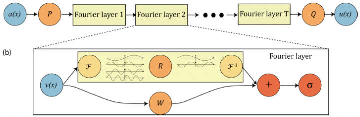
- Pointwise evaluations with Sinc basis



- 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 ?

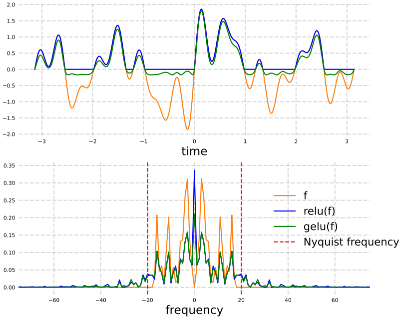


- Convolution in Fourier space $\mathcal{K}$ + Nonlinearity σ
- $\mathcal{K}$ is ReNO wrt Periodic Bandlimited functions $\mathcal{P}_K$:

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

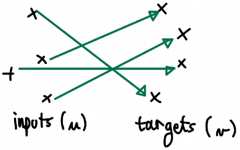
What about activations ?

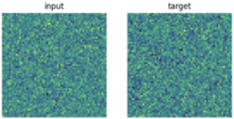
- Nonlinear activation σ can break bandlimits: $\sigma(f) \notin \mathcal{P}_K$
- FNOs are not necessarily ReNOs !!



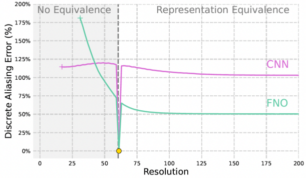
A Synthetic Example: Random Assignment

- The underlying Operator:

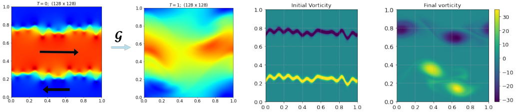




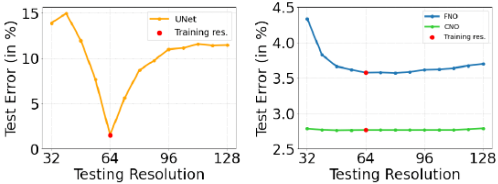
- Errors:



A Practical Example



- FNO Results:



- Challenge: Design a ReNO