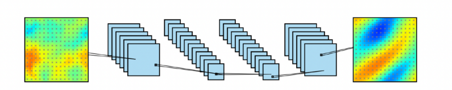
AI in the Sciences and Engineering 2024: Lecture 12

Siddhartha Mishra

Seminar for Applied Mathematics (SAM), D-MATH (and),
ETH AI Center,
ETH Zürich, Switzerland.

What you learnt so far

- Operator learning: Abstract PDE: $\mathcal{D}_a(u) = f$
- Solution Operator: $\mathcal{G} : \mathcal{X} \mapsto \mathcal{Y}$ with $\mathcal{G}(a, f) = u$
- Parametrizations doesn't work in general.
- Uniform Sampling $\mapsto$ CNN $\mapsto$ Interpolation



- Need some notion of Continuous-Discrete Equivalence

ReNO

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

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

- ReNO requires no Aliasing Error: $\varepsilon(\mathcal{G}, G) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E} \equiv 0$
- Leads to a natural form of Resolution Invariance

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

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

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

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

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

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

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:

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

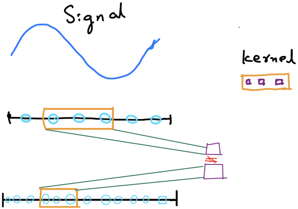
- Nyquist-Shannon $\Rightarrow$ bijection between $\mathcal{X}, X$ on sufficiently dense grid.
- Classical Aliasing Error: $\varepsilon(\mathcal{G}, G) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E}$

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:

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

Alternative: 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:

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

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

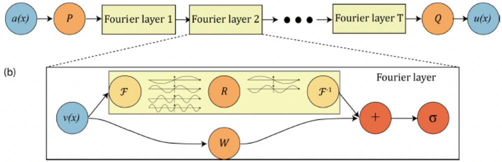
$$(\mathcal{F}^{-1}w_k)(x) = \sum_{k\in\mathbb{Z}^d} w_k\Psi_k(x)$$

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.
- Fast implementation through FFT



Further details on FNOs ?

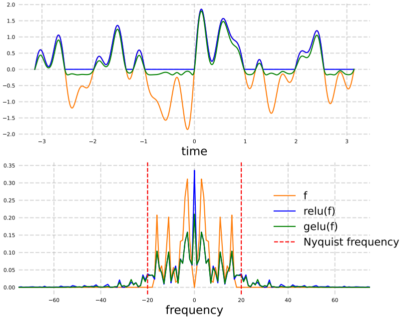
More on FNOs

- FNOs are very widely used in practice !!
- but are FNOs ReNOs ?
- Convolution in Fourier space $\mathcal{K}$ + Nonlinearity σ
- $\mathcal{K}$ is ReNO wrt Periodic Bandlimited functions $\mathcal{P}_K$:

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

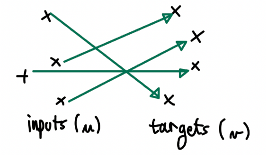
What about activations ?

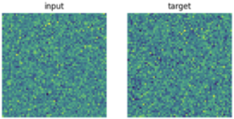
- Nonlinear activation σ can break bandlimits: $\sigma(f) \notin \mathcal{P}_K$
- FNOs are not necessarily ReNOs !!



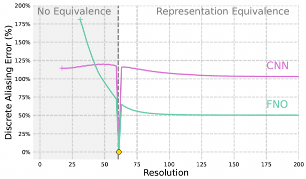
A Synthetic Example: Random Assignment

- The underlying Operator:

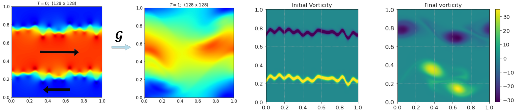




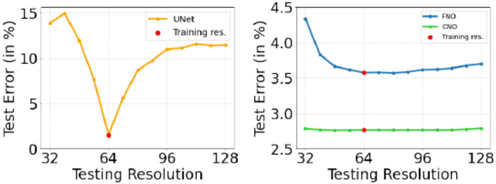
- Errors:



A Practical Example



- FNO Results:



- Challenge: Design a ReNO