# AI in the Sciences and Engineering HS 2026: 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.

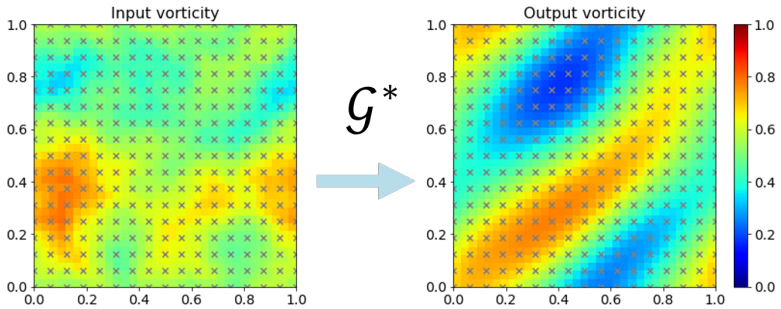
# What we have learnt so far ?

- 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 $\mathcal{G}_{\#}\mu$

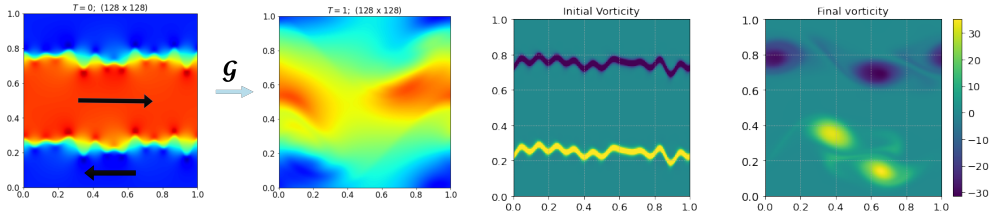
# Solution I: Discretize, then Learn !!




- Discretization $\mapsto$ MLP/CNN $\mapsto$ 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: $\mathcal{G}^* = \mathcal{R} \circ \mathcal{L}^* \circ \mathcal{E}$

# 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 $\infty$-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 $B_\ell, K_\ell$

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

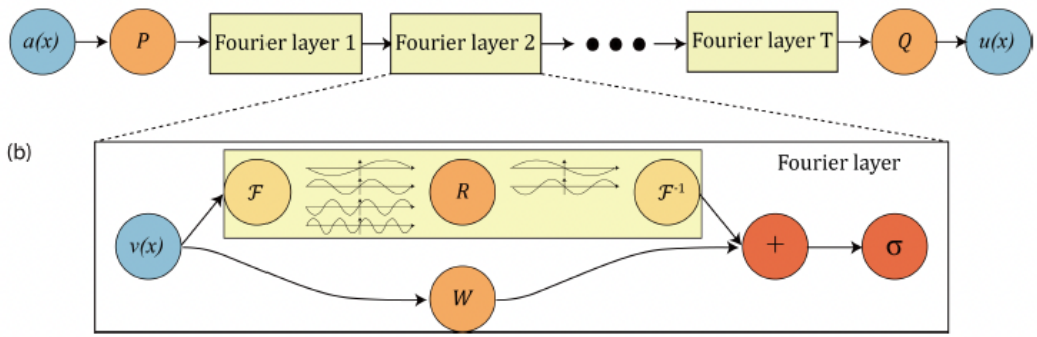
$$(\mathcal{F}^{-1}w_k)(x) = \sum_{k\in\mathbb{Z}^d} w_k\Psi_k(x)$$

# FNO Details

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

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

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



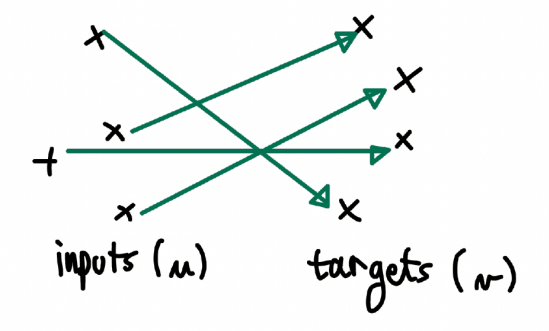
# Theory for FNOs

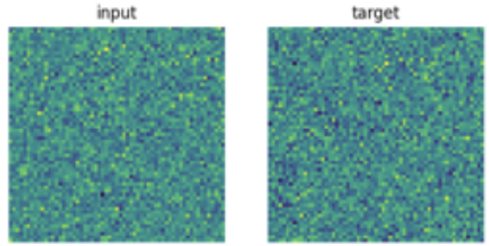
- FNO is very widely used in Practice.
- Theoretical Results of Kovachki, Lanthaler,SM, 2021
- Universal Approximation Thm: For $\mu \in Prob(L^2(D))$ and any measurable $\mathcal{G} : H^r \mapsto H^s$ and $\epsilon > 0$, $\exists \mathcal{N}$ (FNO): $\hat{\mathcal{E}} < \epsilon$
- With Error:

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

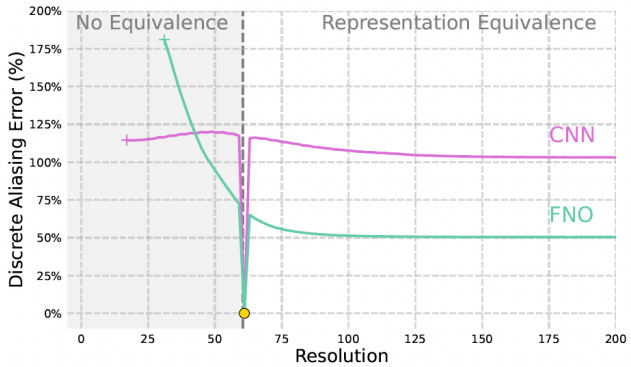
# Is there a Catch ?

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

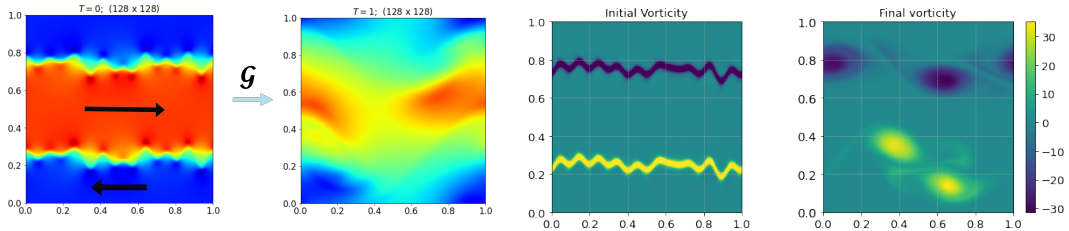


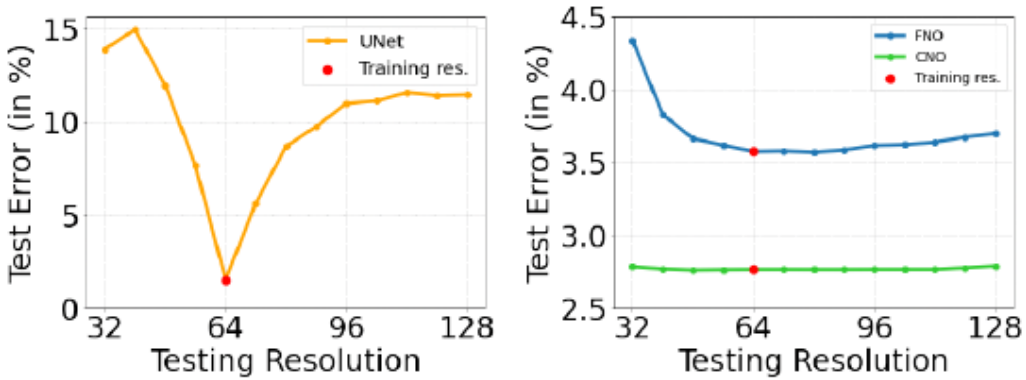

- Errors:



# A Practical Example



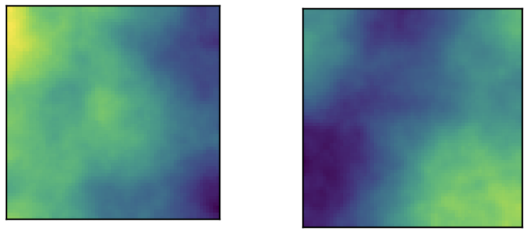

- FNO Results:




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

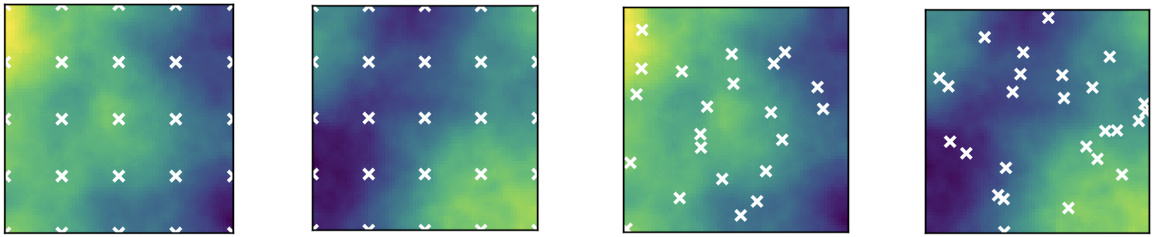
# Why is this a challenge ?

- In principle, Operator maps functions to functions.



Input Output

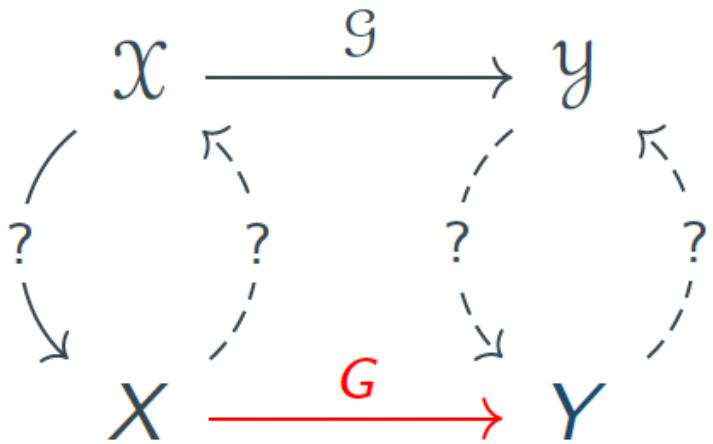
- In practice, both inputs and outputs are Discrete



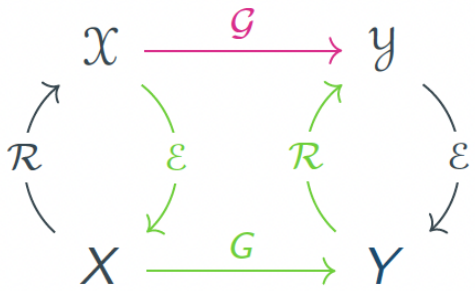
Input Output Input Output

- 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...)




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

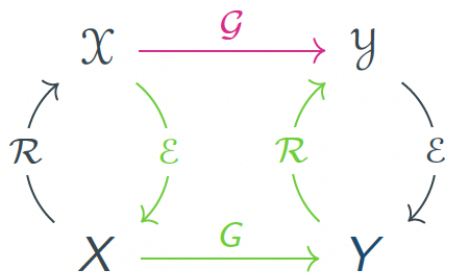
- Discretize, then Learn $\Leftrightarrow$ Learn, then Discretize
- Following Bartolucci et al SM, 2023
- Aliasing error: $\varepsilon(\mathcal{G}, G) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E}$
- Representation Equivalent Neural Operator alias ReNO:

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

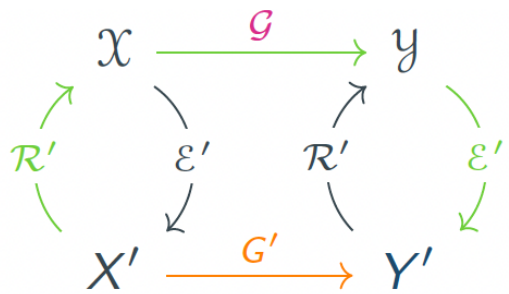
- Concept is instantiated Layerwise: $\mathcal{G} = \mathcal{G}_L \circ \cdots \mathcal{G}_\ell \cdots \mathcal{G}_1$:

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

# A ReNO on different Grids

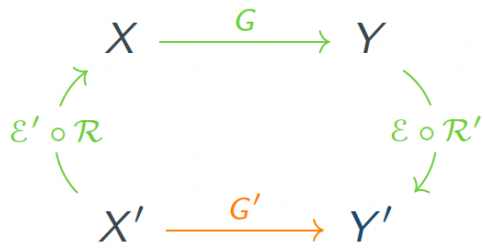



$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E}$$



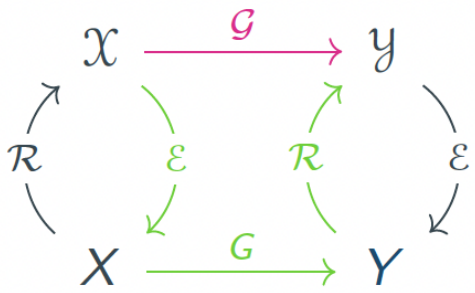

$$G' = \mathcal{E}' \circ \mathcal{G} \circ \mathcal{R}'$$

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




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

# A Concrete Example: 1-D on a Regular Grid




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

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

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

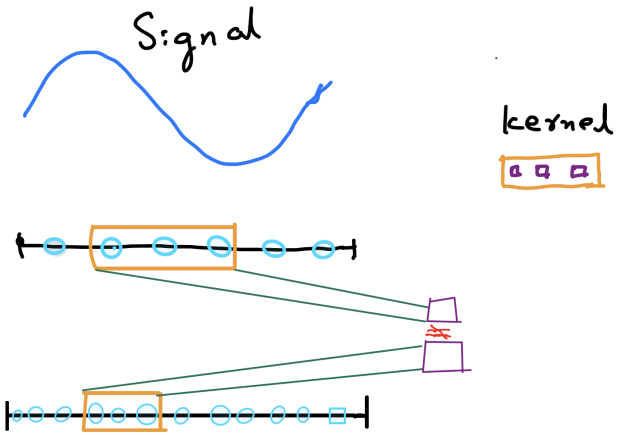
- Nyquist-Shannon $\Rightarrow$ bijection between $\mathcal{X}, X$ on sufficiently dense grid.
- Classical Aliasing Error: $\varepsilon(\mathcal{G}, G) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E}$

# CNNs are not ReNOs !

- CNNs rely on Discrete Convolutions with fixed Kernel:

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

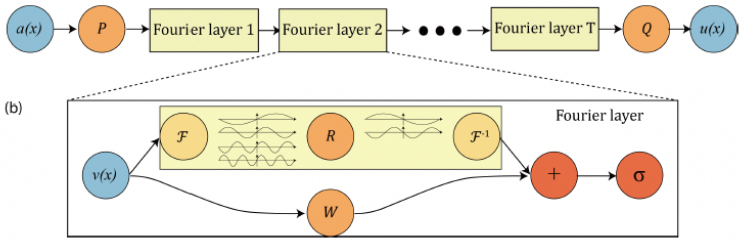
- Pointwise evaluations with Sinc basis




- 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 ?

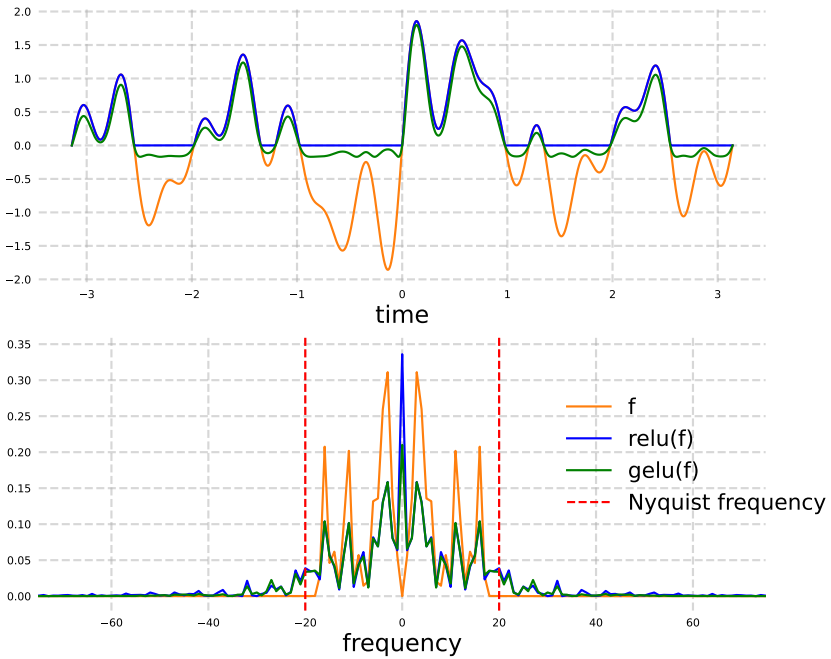



- Convolution in Fourier space $\mathcal{K}$ + Nonlinearity $\sigma$
- $\mathcal{K}$ is ReNO wrt Periodic Bandlimited functions $\mathcal{P}_K$:

$$\begin{array}{ccccccc} \mathcal{P}_K & \xrightarrow{\mathcal{F}} & \mathbb{C}^{2K+1} & \xrightarrow{R\odot} & \mathbb{C}^{2K'+1} & \xrightarrow{\mathcal{F}^{-1}} & \mathcal{P}_{K'} \\ \downarrow{\scriptstyle T^{\dagger}_{\Psi_K}} & & \updownarrow{\scriptstyle \mathrm{Id}} & & \updownarrow{\scriptstyle \mathrm{Id}} & & \uparrow{\scriptstyle T_{\Psi_{K'}}} \\ \mathbb{C}^{2K+1} & \xrightarrow{(2K+1)\cdot \mathrm{F}} & \mathbb{C}^{2K+1} & \xrightarrow{R\odot} & \mathbb{C}^{2K'+1} & \xrightarrow{\frac{1}{(2K'+1)}\cdot \mathrm{F}^{-1}} & \mathbb{C}^{2K'+1} \end{array}$$

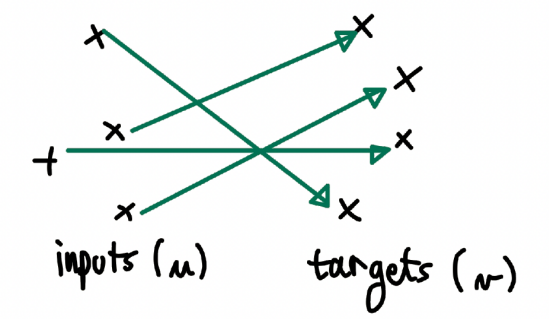
# What about activations ?

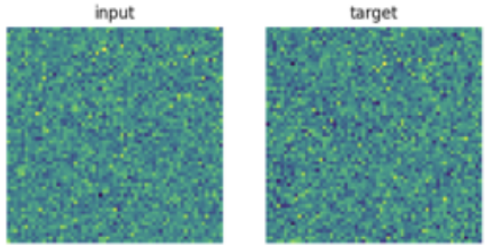
- Nonlinear activation $\sigma$ can break bandlimits: $\sigma(f) \notin \mathcal{P}_K$
- FNOs are not necessarily ReNOs !!



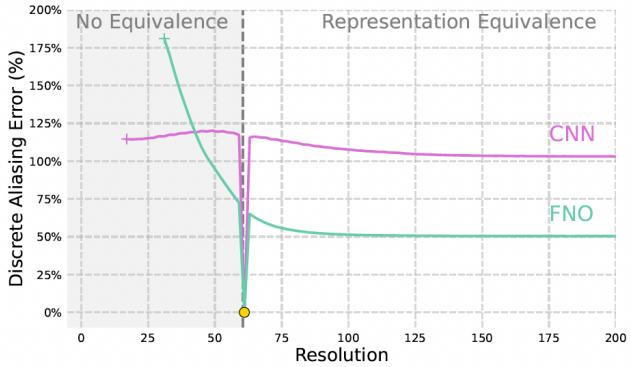
# A Synthetic Example: Random Assignment

- The underlying Operator:

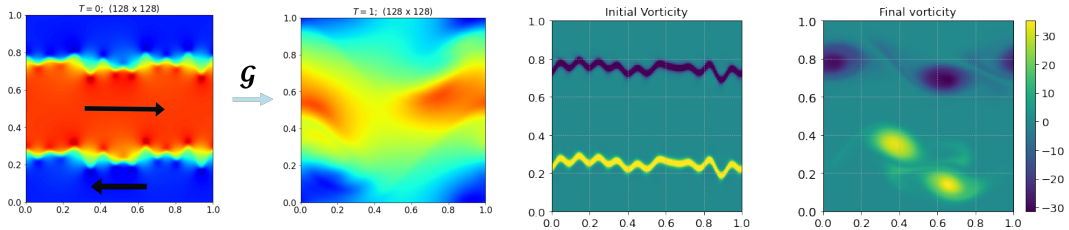


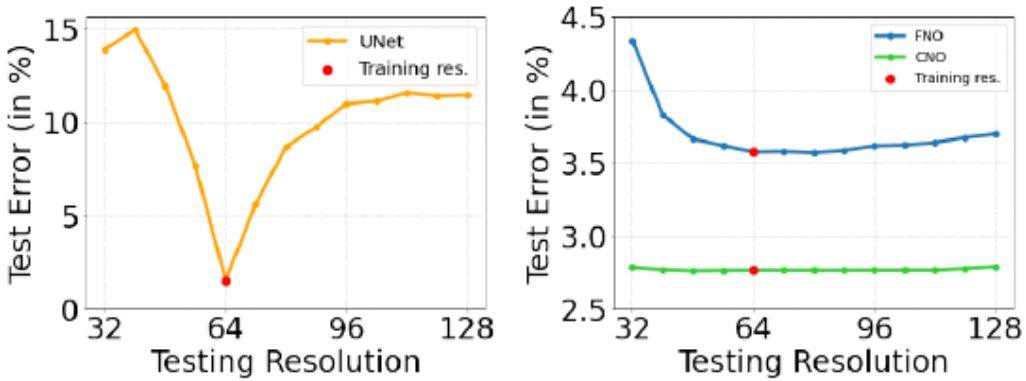

- Errors:



# A Practical Example



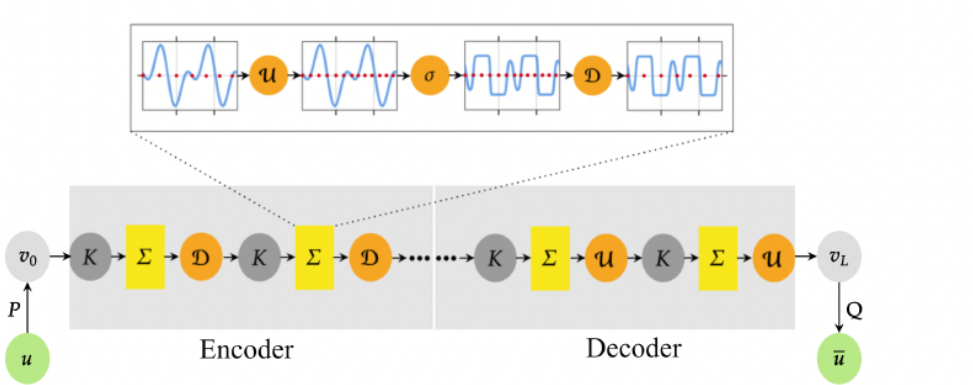

- FNO Results:




- Challenge: Design a ReNO

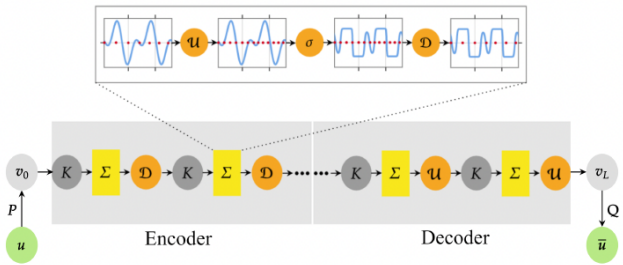
# Convolutions Strike Back !!

- 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.

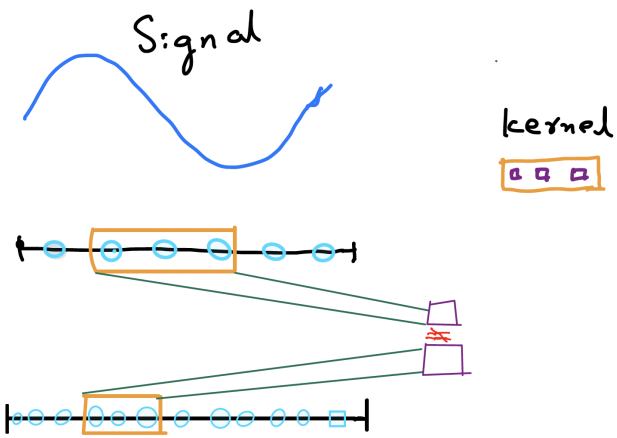
$$\begin{array}{ccc} \mathcal{B}_w & \xrightarrow{\mathcal{K}_w} & \mathcal{B}_w \\ {\scriptstyle T_{\Psi_w}}\uparrow & & \downarrow{\scriptstyle T^*_{\Psi_w}} \\ \ell^2(\mathbb{Z}^2) & \longrightarrow & \ell^2(\mathbb{Z}^2) \end{array}$$

# Contrast with CNNs

- CNNs rely on Discrete Convolutions with fixed Kernel:

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

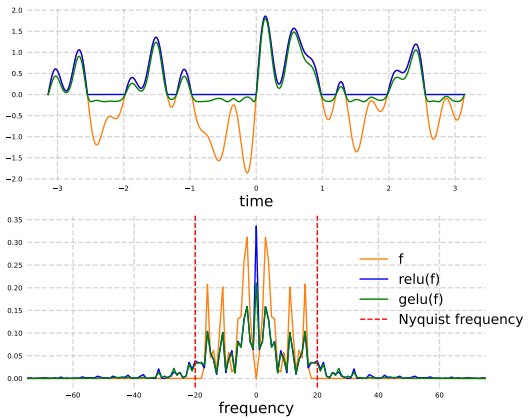
- Pointwise evaluations with Sinc basis

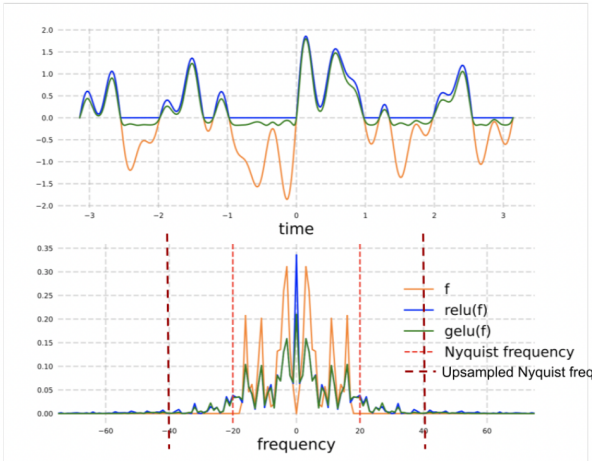



- Easy to check that CNNs are Resolution dependent as:

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

# CNO Key Building Block II: Activation Function ?

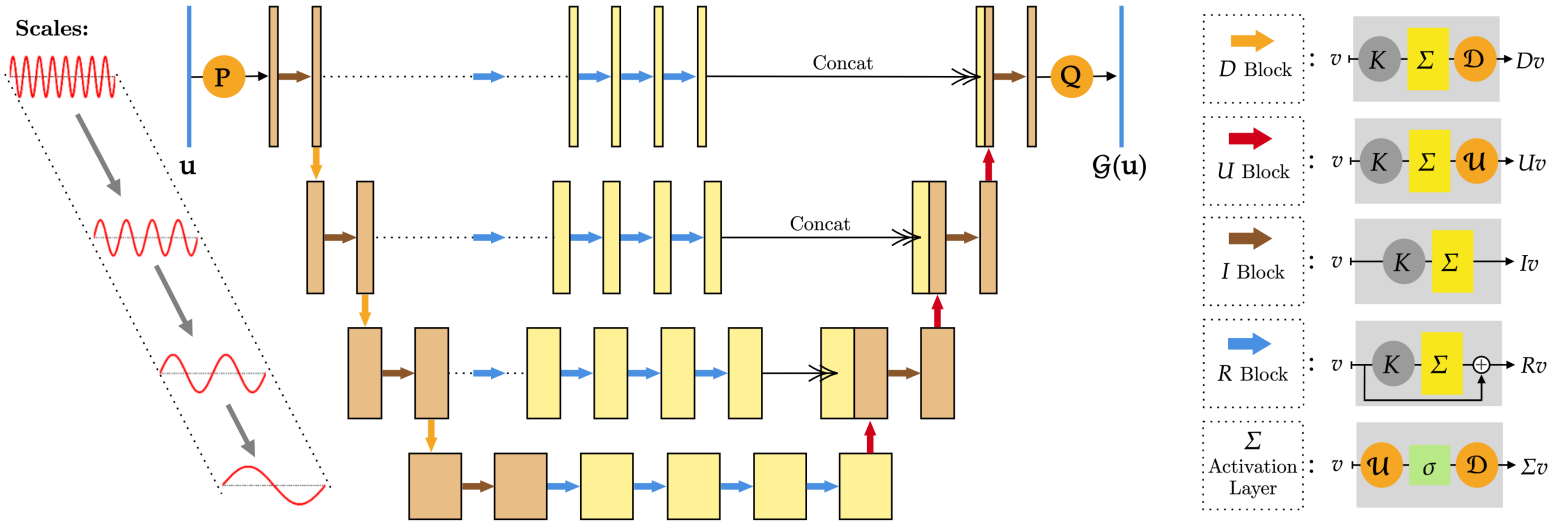



- Apply Activation as $\Sigma : B_w \mapsto B_w$ with $\Sigma = \mathcal{D}_{\bar{w},w} \circ \sigma \circ \mathcal{U}_{w,\bar{w}}$
- Upsampling: $\mathcal{U}_{w,\bar{w}} f = f$ with $w < \bar{w}$
- Downsampling: $\mathcal{D}_{\bar{w},w} f(x) = \left(\frac{\bar{w}}{w}\right)^d \int_D \mathrm{sinc}(2\bar{w}(x-y))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

$$\begin{array}{ccccccc} \mathcal{B}_w & \xrightarrow{P_{\mathcal{B}_{\overline{w}}(\mathbb{R}^2)}} & \mathcal{B}_{\overline{w}} & \xrightarrow{\sigma} & \mathcal{B}_{\overline{w}} & \xrightarrow{P_{\mathcal{B}_w(\mathbb{R}^2)}} & \mathcal{B}_w \\ \uparrow T_{\Psi_w} & & \downarrow T^*_{\Psi_{\overline{w}}} & & \uparrow T_{\Psi_{\overline{w}}} & & \downarrow T^*_{\Psi_w} \\ \ell^2(\mathbb{Z}^2) & \xrightarrow{\mathcal{U}_{w,\overline{w}}} & \ell^2(\mathbb{Z}^2) & \xrightarrow{\sigma} & \ell^2(\mathbb{Z}^2) & \xrightarrow{\mathcal{D}_{\overline{w},w}} & \ell^2(\mathbb{Z}^2) \end{array}$$

# 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 $\mathcal{G} \approx \mathcal{G}^* : \mathcal{B}_w \mapsto \mathcal{B}_{w'}$

$$\begin{array}{ccc} \mathcal{B}_w & \xrightarrow{\mathcal{G}^*} & \mathcal{B}_{w'}, \\ \downarrow T^*_{\Psi_w} & & T_{\Psi_{w'}} \uparrow \\ \ell^2(\mathbb{Z}^2) & \xrightarrow{\mathfrak{G}_{\Psi_w, \Psi_{w'}}} & \ell^2(\mathbb{Z}^2) \end{array}$$

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