REPRESENTATION EQUIVALENT NEURAL OPERATORS

emmanuel de bézenac, ETH Zurich

in collaboration with

bogdan raonić

francesca bartolucci

roberto molinaro

tobias rohner

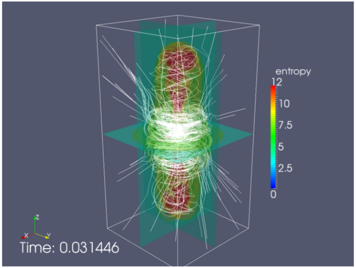
tim de ryck

rima alaifari

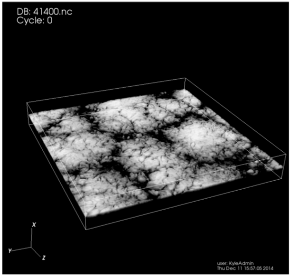
siddhartha mishra

Outline of the presentation

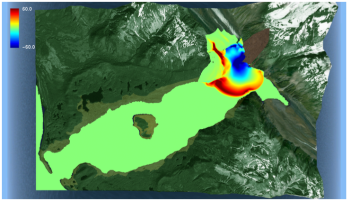
- What are neural operators for solving PDEs ?
- Are neural operators really operators ?
- How can we correctly define neural operators ?
- Can construct practical ones ?



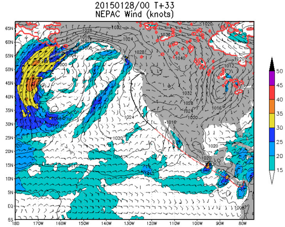
Supernovas



Clouds



Tsunamis



Weather

Solving PDEs as approximating solution operators

Theory

- We want to solve the PDE :

$$\mathcal{F}(u, p) = 0,\ x \in \Omega$$
$$B(u, p) = 0,\ x \in \partial\Omega$$

- p input, e.g. initial condition

Solving PDEs as approximating solution operators

Theory

- We want to solve the PDE :

$$\mathcal{F}(u, p) = 0, \ x \in \Omega$$
$$B(u, p) = 0, \ x \in \partial\Omega$$

- p input, e.g. initial condition
- u, p lie in a (potentially **infinite dim**) **function spaces** $\mathcal{X}, \mathcal{Y}$

Theory

- We want to solve the PDE :

$$\mathcal{F}(u, p) = 0, \; x \in \Omega$$
$$B(u, p) = 0, \; x \in \partial\Omega$$

- p input, e.g. initial condition
- u, p lie in a (potentially **infinite dim**) **function spaces** $\mathcal{X}, \mathcal{Y}$
- consider solution operator $\mathcal{G}$

$$\mathcal{G} : \mathcal{X} \to \mathcal{Y},$$
$$p \to u$$

Solving PDEs as approximating solution operators

Theory

- We want to solve the PDE :

$$\mathcal{F}(u, p) = 0,\ x \in \Omega$$
$$B(u, p) = 0,\ x \in \partial\Omega$$

- p input, e.g. initial condition
- u, p lie in a (potentially **infinite dim**) **function spaces** $\mathcal{X}, \mathcal{Y}$
- consider solution operator $\mathcal{G}$

$$\mathcal{G} : \mathcal{X} \to \mathcal{Y},$$
$$p \to u$$

Practice

- Only finite number of computations,

Theory

- We want to solve the PDE :

$$\mathcal{F}(u, p) = 0, \ x \in \Omega$$
$$B(u, p) = 0, \ x \in \partial\Omega$$

- p input, e.g. initial condition
- u, p lie in a (potentially **infinite dim**) **function spaces** $\mathcal{X}, \mathcal{Y}$
- consider solution operator $\mathcal{G}$

$$\mathcal{G} : \mathcal{X} \to \mathcal{Y},$$
$$p \to u$$

Practice

- Only finite number of computations,
- need to discretize u, p e.g. on a regular grid Δ

Theory

- We want to solve the PDE :

$$\mathcal{F}(u, p) = 0, \ x \in \Omega$$
$$B(u, p) = 0, \ x \in \partial\Omega$$

- p input, e.g. initial condition
- u, p lie in a (potentially **infinite dim**) **function spaces** $\mathcal{X}, \mathcal{Y}$
- consider solution operator $\mathcal{G}$

$$\mathcal{G} : \mathcal{X} \to \mathcal{Y},$$
$$p \to u$$

Practice

- Only finite number of computations,
- need to discretize u, p e.g. on a regular grid Δ
- consider discrete spaces X, Y

Theory

- We want to solve the PDE :

$$\mathcal{F}(u,p) = 0, \ x \in \Omega$$
$$B(u,p) = 0, \ x \in \partial\Omega$$

- p input, e.g. initial condition
- u, p lie in a (potentially **infinite dim**) **function spaces** $\mathcal{X}, \mathcal{Y}$
- consider solution operator $\mathcal{G}$

$$\mathcal{G} : \mathcal{X} \to \mathcal{Y},$$
$$p \to u$$

Practice

- Only finite number of computations,
- need to discretize u, p e.g. on a regular grid Δ
- consider discrete spaces X, Y
- consider approximation G

$$G : X \to Y,$$
$$p_\Delta \to u_\Delta$$

Link between discrete and continuous

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, dashed, bend right, "?"'] & \mathcal{Y} \arrow[d, dashed, bend right, "?"'] \\
X \arrow[r, "G"] \arrow[u, dashed, bend right, "?"'] & Y \arrow[u, dashed, bend right, "?"']
\end{tikzcd}$$

Having an **operator** perspective is very important :

- the solution lies in this function space,
- quantifying discrepancy between discrete and continuous solutions is at the basis of numerical analysis.
- Essential for structure preserving methods, as symmetries are at the operator level, etc...

For this reason,

- solution operator $\mathcal{G}$ approximated with **neural operator** $\mathcal{G}_\theta$,
- $\mathcal{G}_\theta$ mapping from functions to functions :

$$\begin{aligned} \mathcal{G}_\theta : \mathcal{X} &\to \mathcal{Y}, \\ p &\to u \end{aligned}$$

- potential for deep learning to drastically *accelerate* simulations,
- can be applied even when pde is unknown (e.g. climate)

Discretization invariance

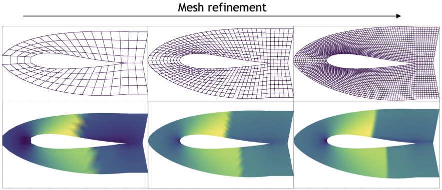
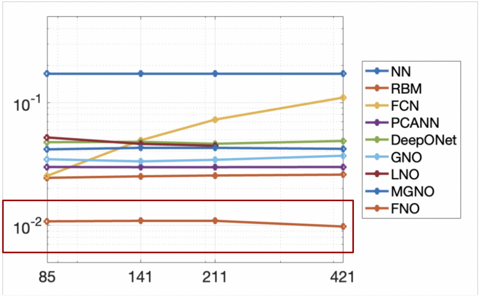


Figure 1: Discretization Invariance
An discretization-invariant operator has convergent predictions on a mesh refinement.



a prototypical example : the Fourier neural operator (fno)

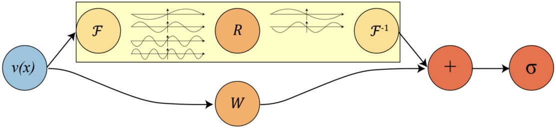


Figure 2 – layer of the Fourier neural operator **?**

- sequence of layers, as in classical nns

$$\mathcal{G} = \mathcal{N}_L \circ \mathcal{N}_{L-1} \circ \ldots \circ \mathcal{N}_1$$
$$(\mathcal{N}_\ell v)(x) = \sigma\left(A_\ell v(x) + B_\ell(x) + \mathcal{K}_\ell v(x)\right)$$

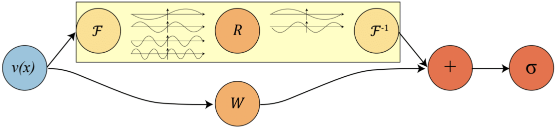


Figure 2 – layer of the Fourier neural operator **?**

- sequence of layers, as in classical nns

$$\mathcal{G} = \mathcal{N}_L \circ \mathcal{N}_{L-1} \circ \ldots \circ \mathcal{N}_1$$
$$(\mathcal{N}_\ell v)(x) = \sigma\left(A_\ell v(x) + B_\ell(x) + \mathcal{K}_\ell v(x)\right)$$

- **novelty** : layers defined as mappings from functions to functions

a prototypical example : the Fourier neural operator (fno)

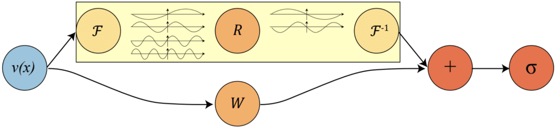


Figure 3 – layer of the Fourier neural operator

- for example, continuous convolution with Fourier layer :

$$\mathcal{K}_\ell v(x) = \int_D k_\ell(x-y)v(y)dy = \mathfrak{F}^{-1}(\mathcal{R}_\theta \odot \mathfrak{F}(v))(x)$$

claim : gives ability to handle different resolutions

a prototypical example : the Fourier neural operator (fno)

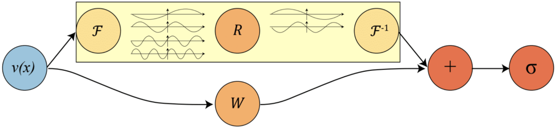


Figure 3 – layer of the Fourier neural operator

- for example, continuous convolution with Fourier layer :

$$\mathcal{K}_\ell v(x) = \int_D k_\ell(x-y)v(y)dy = \mathcal{F}^{-1}(\mathcal{R}_\theta \odot \mathcal{F}(v))(x)$$

  claim : gives ability to handle different resolutions

- state of the art for many PDEs, claims to handle functions
- claim to be discretization invariant,
- how are these computed on a computer ?

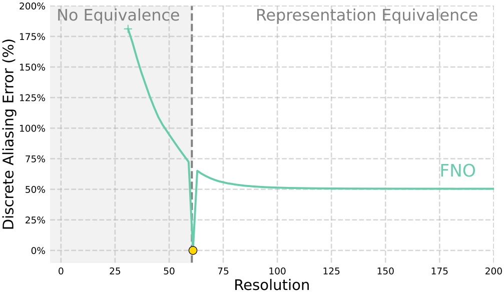
FNO in practice

$$
\begin{tikzcd}
\mathcal{X} \arrow[r, "\mathcal{G}"] \arrow[d, bend right, "?"'] & \mathcal{Y} \arrow[d, bend right, dashed, "?"'] \\
X \arrow[r, red, "G"] \arrow[u, bend right, dashed, "?"'] & Y \arrow[u, bend right, dashed, "?"']
\end{tikzcd}
$$

- in practice, all computations done discretely using G, not $\mathcal{G}$
  - input and output function sampled on a grid
  - DFT is used instead of Fourier transform
  - activations computed on the grid (not the function)

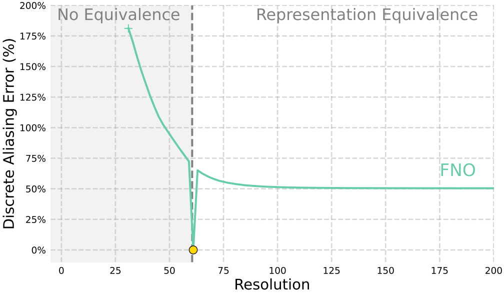
- link between operator $\mathcal{G}$ and discrete map G ?

Discrete representations not equivalent



- random input $u \in \mathbb{R}^{61}$ and target data $v \in \mathbb{R}^{61}$, $u_i, v_i \sim \mathcal{G}(0, 1)$
- train mapping G to regress $u \to v$
- after training to 0 loss, change resolution and compute discrepancy

Discrete representations not equivalent



- random input $u \in \mathbb{R}^{61}$ and target data $v \in \mathbb{R}^{61}$, $u_i, v_i \sim \mathcal{G}(0, 1)$
- train mapping G to regress $u \to v$
- after training to 0 loss, change resolution and compute discrepancy

- hints at discrepancy between continuous and discrete
- a bit of an extreme case, but shows that
- discretization invariance is only a property at the limit, does not say anything for practical resolutions

Representation equivalent neural operators (ReNO)

Representation equivalent neural operators (ReNO)

Couple $(G, \mathcal{G})$, such that the diagram commutes :

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

Representation equivalent neural operators (ReNO)

Representation equivalent neural operators (ReNO)

Couple $(G, \mathcal{G})$, such that the diagram commutes :

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

- i.e. discrepancy between continuous and discrete, aka *aliasing*

$$\varepsilon(G, \mathcal{G}) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E} = 0,$$

Representation equivalent neural operators (ReNO)

Representation equivalent neural operators (ReNO)

Couple $(G, \mathcal{G})$, such that the diagram commutes :

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

- i.e. discrepancy between continuous and discrete, aka *aliasing*

$$\varepsilon(G, \mathcal{G}) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E} = 0,$$

- $\mathcal{X}, \mathcal{Y}$ separable Hilbert spaces, e.g. bandlimited functions, spanned by wavelets, fourier series coefficients, etc...

layer-wise Instantiation

- Design each layer $(\mathcal{G}_\ell, G_\ell)$ such that diagram **commutes** :

$$\mathcal{G}_\ell = \mathcal{R} \circ N_\ell \circ \mathcal{E}$$

- composition of layers is also a ReNO

Equivalence between discrete representations

Consider two discretizations G and G' of $\mathcal{G}$ (e.g. on different grids).

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

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

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

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

If both diagrams commute, **discrete representations are equivalent** :

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

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

Equivalence between discrete representations

Consider two discretizations G and G' of $\mathcal{G}$ (e.g. on different grids).

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

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

If both diagrams commute, **discrete representations are equivalent** :

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

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

- If **not** $\varepsilon(G, \mathcal{G}) \neq 0$, potential **discrepancy** at different resolutions.

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

Instantiation : Bandlimited spaces

- input, output spaces : $\mathcal{X}, \mathcal{Y}$ *bandlimited* functions,

$$\mathcal{E}(v) = \{f(x_i)\}_{1,\dots,n}, \qquad \mathcal{R}(v)(x) = \sum_{i=1}^{n} v(x_i)\mathrm{sinc}(x - x_i)$$

- Natural spaces for point-wise evaluations on cartesian grid,

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

Instantiation : Bandlimited spaces

- input, output spaces : $\mathcal{X}, \mathcal{Y}$ *bandlimited* functions,

$$\mathcal{E}(v) = \{f(x_i)\}_{1,\ldots,n}, \qquad \mathcal{R}(v)(x) = \sum_{i=1}^{n} v(x_i)\mathrm{sinc}(x - x_i)$$

- Natural spaces for point-wise evaluations on cartesian grid,
- Nyquist-Shannon : if grid dense enough, bijection between $\mathcal{X}$ and X

Instantiation

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

Instantiation : Bandlimited spaces

- input, output spaces : $\mathcal{X}, \mathcal{Y}$ *bandlimited* functions,

$$\mathcal{E}(v) = \{f(x_i)\}_{1,\dots,n}, \qquad \mathcal{R}(v)(x) = \sum_{i=1}^{n} v(x_i)\mathrm{sinc}(x - x_i)$$

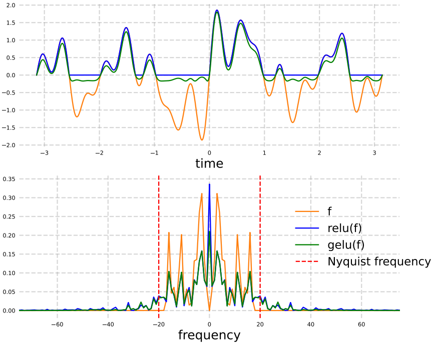
- Natural spaces for point-wise evaluations on cartesian grid,
- Nyquist-Shannon : if grid dense enough, bijection between $\mathcal{X}$ and X
- $\varepsilon(G, \mathcal{G}) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E}$ reduces to classical aliasing.

per layer analysis

- Continuous and discrete convolutions **equivalent**,

per layer analysis

- Continuous and discrete convolutions **equivalent**,
- Activation function is not : $\varepsilon(G, \mathcal{G}) \neq 0$, i.e. diagram does not commute



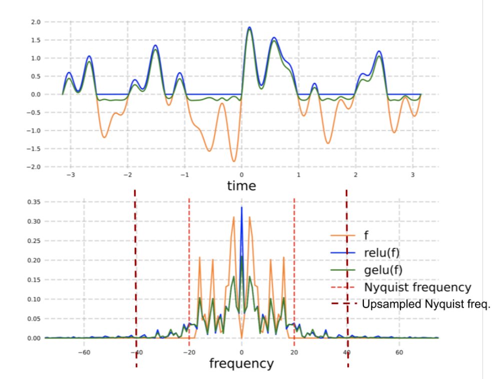
New convolutional based architecture

We propose to

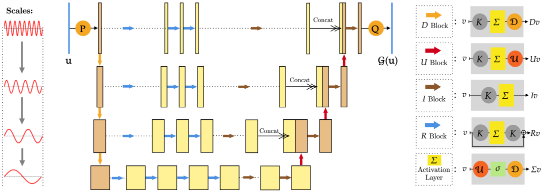
- *Fix* the activation function,
- Go back to standard convolutions (Fourier not essential),

Just as in StyleGAN3, new activation :

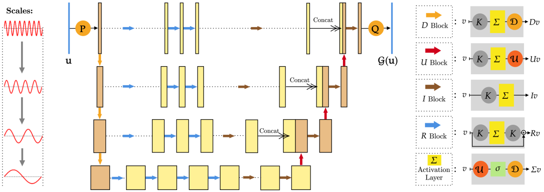
- upsample
- activation
- downsample



a convolutional based neural operator (CNO)

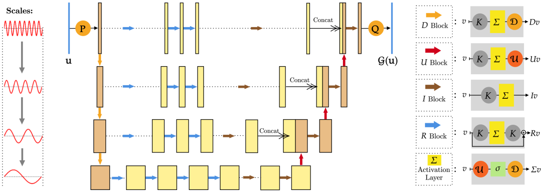


a convolutional based neural operator (CNO)



- convolution operation, implemented with simple convs :

$$K_\ell = \sum_{i,j=1}^{k} k_{ij} \cdot \delta_{z_{ij}}, \qquad \mathcal{K}_\ell v(x) = \int_D K_\ell(x-y)v(y)dy$$



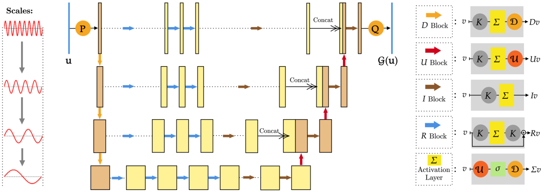
- convolution operation, implemented with simple convs :

$$K_\ell = \sum_{i,j=1}^{k} k_{ij} \cdot \delta_{z_{ij}}, \qquad \mathcal{K}_\ell v(x) = \int_D K_\ell(x-y)v(y)dy$$

- activation as in StyleGAN3 with up/down sampling

$$\Sigma(f)(x) = \mathcal{D} \circ \sigma \circ \mathcal{U}$$

a convolutional based neural operator (CNO)



- convolution operation, implemented with simple convs :

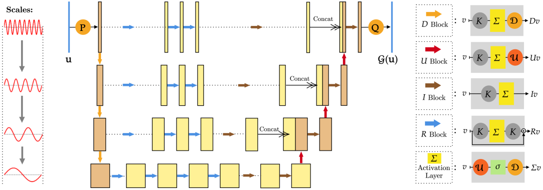
$$K_\ell = \sum_{i,j=1}^{k} k_{ij} \cdot \delta_{z_{ij}}, \qquad \mathcal{K}_\ell v(x) = \int_D K_\ell(x-y)v(y)dy$$

- activation as in StyleGAN3 with up/down sampling

$$\Sigma(f)(x) = \mathcal{D} \circ \sigma \circ \mathcal{U}$$

- upsampling $\mathcal{U}$ and downsampling $\mathcal{D}$: interpolation with sinc filter

a convolutional based neural operator (CNO)



- convolution operation, implemented with simple convs :

$$K_\ell = \sum_{i,j=1}^{k} k_{ij} \cdot \delta_{z_{ij}}, \qquad \mathcal{K}_\ell v(x) = \int_D K_\ell(x-y)v(y)dy$$

- activation as in StyleGAN3 with up/down sampling

$$\Sigma(f)(x) = \mathcal{D} \circ \sigma \circ \mathcal{U}$$

- upsampling $\mathcal{U}$ and downsampling $\mathcal{D}$: interpolation with sinc filter
- these operations all have a (unique) discrete N_ℓ (approximately)

$$\mathcal{G}_\ell(v) = \mathcal{R} \circ N_\ell \circ \mathcal{E}(v) \quad \text{for all} \quad v \in \mathcal{X}$$

- preservation of continuous structures, translation equivariance (better generalization ?)
- cno able to process at different grids, not restricted to fno layer
- *representation equivalence*

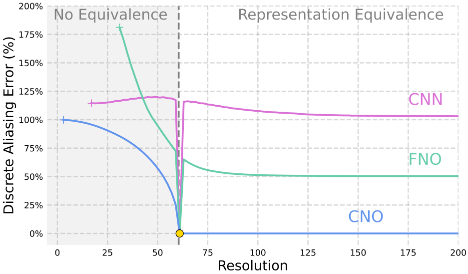


Table 1 – Relative median L^1 test errors.

	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%**

Table 1 – Relative median L^1 test errors.

	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%**

- grid search for all baselines,

quantitative results

Table 1 – Relative median L^1 test errors.

	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%**

- grid search for all baselines,
- convolutional architectures very good across many PDEs,

quantitative results

Table 1 – Relative median L^1 test errors.

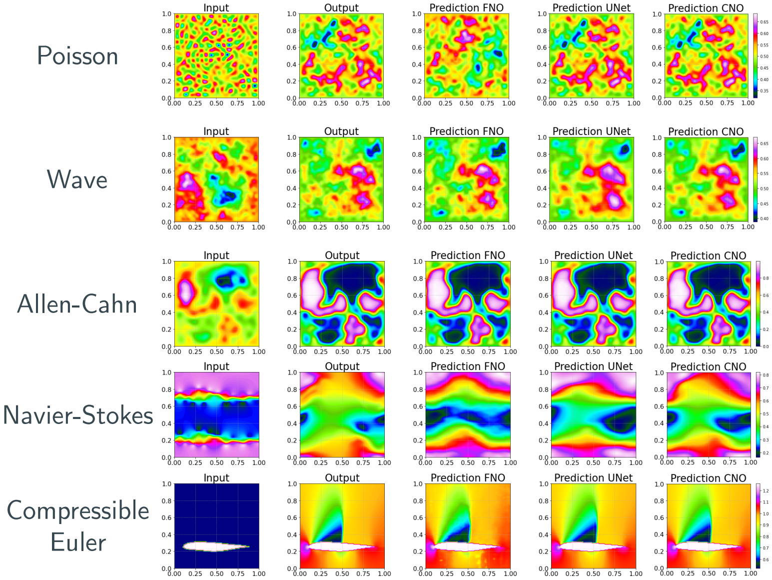
	In/Out	FFNN	GT	UNet	ResNet	DON	FNO	CNO
Poisson Equation	In	5.74%	2.77%	0.71%	0.43%	12.92%	4.98%	**0.21%**
	Out	5.35%	2.84%	1.27%	1.10%	9.15%	7.05%	**0.27%**
Wave Equation	In	2.51%	1.44%	1.51%	0.79%	2.26%	1.02%	**0.63%**
	Out	3.01%	1.79%	2.03%	1.36%	2.83%	1.77%	**1.17%**
Smooth Transport	In	7.09%	0.98%	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%**

- grid search for all baselines,
- convolutional architectures very good across many PDEs,
- if not interested in different grids, and structure preservation, probably a good choice to consider UNets,

Table 1 – Relative median L^1 test errors.

	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%**

- grid search for all baselines,
- convolutional architectures very good across many PDEs,
- if not interested in different grids, and structure preservation, probably a good choice to consider UNets,
- e.g. for Navier Stokes, 10^{3-4} speedups wrt sota GPU codes


Poisson
Wave
Allen-Cahn
Navier-Stokes
Compressible Euler
Input
Output
Prediction FNO
Prediction UNet
Prediction CNO

- highlighted discrepancies between discrete and continuous operations in neural operators,

conclusion

- highlighted discrepancies between discrete and continuous operations in neural operators,
- just as with any numerical method, we should pay attention to errors introduced by discretization,

conclusion

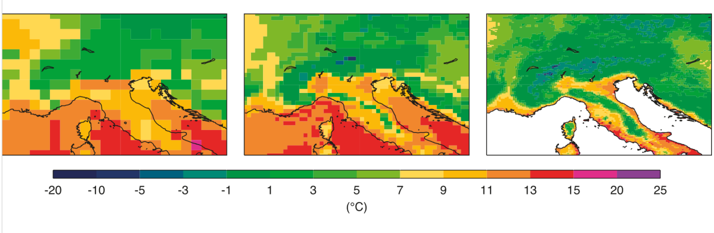
- highlighted discrepancies between discrete and continuous operations in neural operators,
- just as with any numerical method, we should pay attention to errors introduced by discretization,
- we have formulated a framework to take them into account,

conclusion

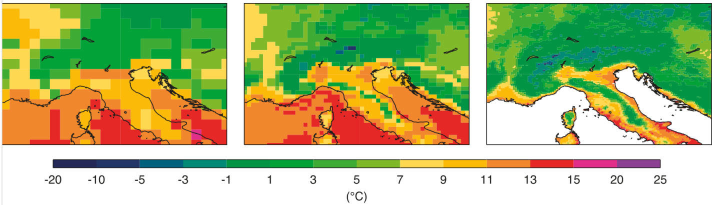
- highlighted discrepancies between discrete and continuous operations in neural operators,
- just as with any numerical method, we should pay attention to errors introduced by discretization,
- we have formulated a framework to take them into account,
- we have proposed an accurate architecture to

conclusion

- highlighted discrepancies between discrete and continuous operations in neural operators,
- just as with any numerical method, we should pay attention to errors introduced by discretization,
- we have formulated a framework to take them into account,
- we have proposed an accurate architecture to
- can effectively link computations at different discretizations



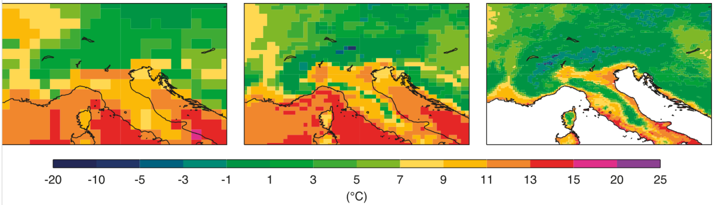
- only considered perfect equivalence between discrete and continuous
  $\rightarrow$ relaxation to $\varepsilon(\mathcal{G}, G) \leq \epsilon$?



- only considered perfect equivalence between discrete and continuous
  $\rightarrow$ relaxation to $\varepsilon(\mathcal{G}, G) \leq \epsilon$?

- climate data often on different grids,
  - often different resolutions
  - there may be structures to preserve, e.g. conservation laws, etc,
  - can we learn one neural network for datasets at different resolutions ?
  - potentially useful to trade accuracy for compute and memory

limitations and future work



- only considered perfect equivalence between discrete and continuous
  $\rightarrow$ relaxation to $\varepsilon(\mathcal{G}, G) \leq \epsilon$?

- climate data often on different grids,
  - often different resolutions
  - there may be structures to preserve, e.g. conservation laws, etc,
  - can we learn one neural network for datasets at different resolutions?
  - potentially useful to trade accuracy for compute and memory

- thanks for your time!

Appendix

Representation equivalent neural operators (ReNO)

Representation equivalent neural operators (ReNO)

Couple $(G, \mathcal{G})$, such that the diagram commutes :

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

- $f = \sum_{i \in I} c_i \Phi_i(x)$. discretize : $f \xrightarrow{\mathcal{E}} c$, reconstruct : $c \xrightarrow{\mathcal{R}} f$
- $\{\Phi_i\}_{i \in I}$ basis or frame spanning $\mathcal{X}, \mathcal{Y}$,
- i.e. discrepancy between continuous and discrete, aka *aliasing*

$$\varepsilon(G, \mathcal{G}) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E} = 0,$$

Representation equivalent neural operators (ReNO)

Representation equivalent neural operators (ReNO)

Couple $(G, \mathcal{G})$, such that the diagram commutes :

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

- $f = \sum_{i \in I} c_i \Phi_i(x)$. discretize : $f \xrightarrow{\mathcal{E}} c$, reconstruct : $c \xrightarrow{\mathcal{R}} f$
- $\{\Phi_i\}_{i \in I}$ basis or frame spanning $\mathcal{X}, \mathcal{Y}$,
- i.e. discrepancy between continuous and discrete, aka *aliasing*

$$\varepsilon(G, \mathcal{G}) = \mathcal{G} - \mathcal{R} \circ G \circ \mathcal{E} = 0,$$

- $\mathcal{X}, \mathcal{Y}$ separable Hilbert spaces, e.g. bandlimited functions, spanned by wavelets, fourier series coefficients, etc...

frame theory

synthesis and analysis operators

- $\mathcal{E}\colon \mathcal{X} \to \ell^2(I), \quad \mathcal{E}(\{f_i\}_{i\in I}) = \{\langle f, S^{-1} f_i\rangle\}_{i\in I}$
- $\mathcal{R}\colon \ell^2(I) \to \mathcal{X}, \quad \mathcal{R}(\{c_i\}_{i\in I}) = \sum_{i\in I} c_i f_i,$
- $S(f) = \sum_{i\in I} \langle f, f_i\rangle f_i$

definition : a frame

A countable sequence of vectors $\{f_i\}_{i\in I}$ in $\mathcal{X}$ is a frame for $\mathcal{X}$ if there exists constants $A, B > 0$ such that for all $f \in \mathcal{X}$

$$A\|f\|^2 \leq \sum_{i\in I} |< f, f_i >|^2 \leq B\|f\|^2$$

- e.g. orthogonal basis :$A = B = 1$,

frame decomposition theorem

Every element in $\mathcal{X}$ can be *uniquely and stably* reconstructed from its frame coefficients by means of the reconstruction formula

$$f = \mathcal{R}\mathcal{E}f = \sum_{i \in I} \langle f, S^{-1} f_i \rangle f_i = \sum_{i \in I} \langle f, f_i \rangle S^{-1} f_i, \tag{1}$$

where the series converge unconditionally.

Ablation Study

Table 13: Relative median L^1 test errors, for both in- and out-of-distribution testing, for the CNO models and two ablation models.

	In/Out	CNO	CNO w/o Filters	CNO w/o ResNets
Poisson Equation	In	**0.21%**	0.93%	0.85%
	Out	**0.27%**	1.65%	0.82%
Wave Equation	In	0.63%	**0.59%**	1.64%
	Out	1.17%	**1.12%**	1.64%
Smooth Transport	In	**0.24%**	0.31%	0.31%
	Out	**0.46%**	**0.46%**	0.76%
Discontinuous Transport	In	**1.03%**	1.21%	1.17%
	Out	**1.18%**	1.32%	1.60%
Allen-Cahn	In	**0.54%**	0.69%	0.71%
	Out	2.23%	**2.16%**	2.21%
Navier-Stokes	In	**2.76%**	3.20%	3.00%
	Out	7.04%	9.60%	**5.85%**
Darcy	In	**0.38%**	0.47%	0.41%
	Out	**0.50%**	0.65%	0.58%
Compressible Euler	In	**0.35%**	0.38%	0.37%
	Out	**0.59%**	0.62%	**0.59%**

Proposition 3.8. *Let (U, u) be an $\epsilon-$ReNO. For any two frame sequence pairs (Ψ, Φ) and (Ψ', Φ') satisfying conditions in Definition 3.4 and such that $\mathcal{M}_{\Phi'} \subseteq \mathcal{M}_{\Phi}$, we have*

$$\|\tau(u, u')\| \leq \frac{2\epsilon\sqrt{B_\Psi}}{\sqrt{A_\Phi}},$$