SOME INSIGHTS INTO NEURAL OPERATORS FOR SOLVING PDES

Emmanuel de Bézenac

Outline of the Presentation

1. From Functions to Operators (and back)
   - A new tool to solve PDEs by learning the solution operator
   - The Fourier Neural Operator (FNO)
   - Understanding what it means to be an operator
   - From FNO to the Convolutional Neural Operator (CNO)

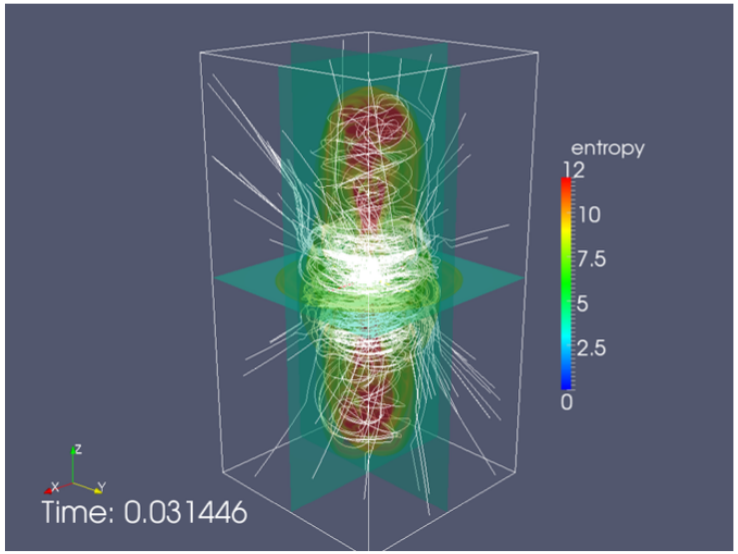
Outline of the Presentation

1. From Functions to Operators (and back)
   - A new tool to solve PDEs by learning the solution operator
   - The Fourier Neural Operator (FNO)
   - Understanding what it means to be an operator
   - From FNO to the Convolutional Neural Operator (CNO)

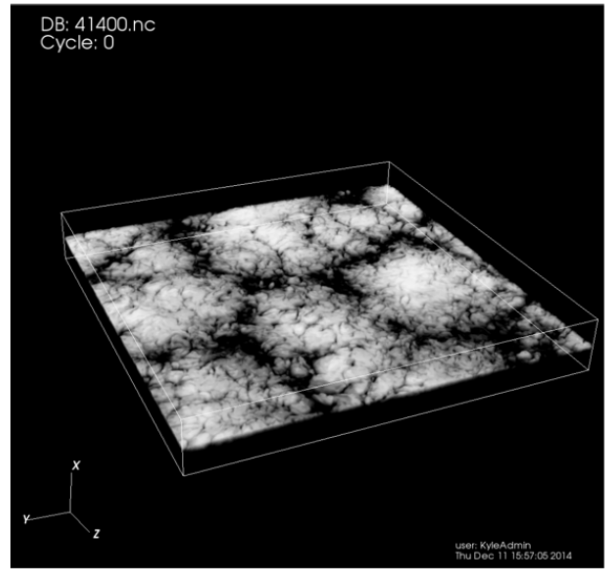
2. From Numerics to Learning
   - Decomposing the sources of error :

     $$\text{Total Error} = \text{Discretization} + \text{Training} + \text{Generalization}$$

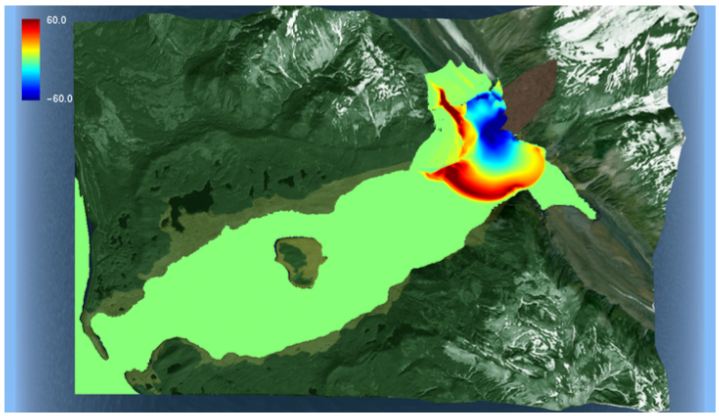
   - Tackling the Generalization error



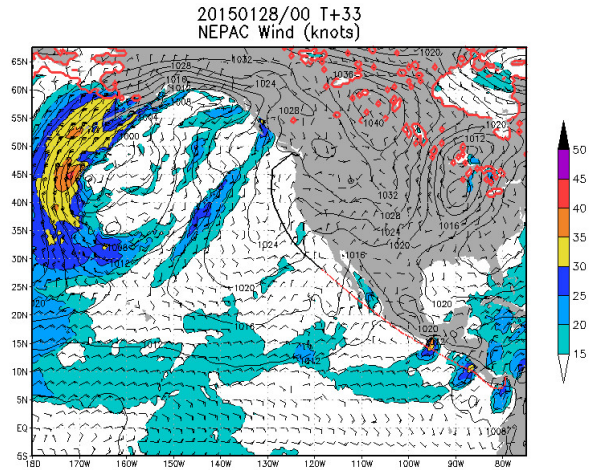
Supernovas



Clouds



Tsunamis



Weather

Solving PDEs as approximating solution operators

- We want to solve the PDE :

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

- p input, e.g. initial condition

Solving PDEs as approximating solution operators

- We want to solve the PDE :

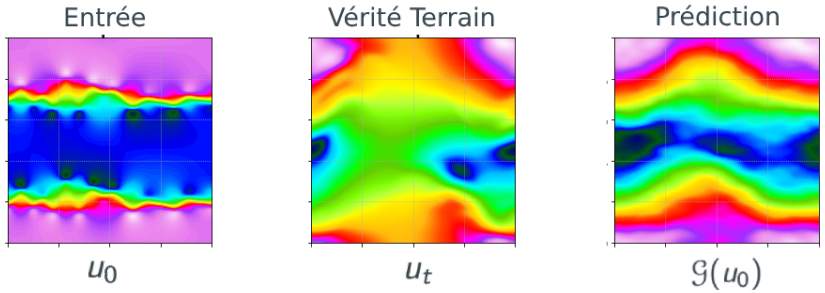
$$\mathcal{D}(u, p) = 0,\ x \in \Omega$$
$$B(u, p) = 0,\ x \in \partial\Omega$$

- p input, e.g. initial condition

- Consider solution operator $\mathcal{G}$

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

Solving PDEs as approximating solution operators



Navier-Stokes solution operator.

- We want to solve the PDE :

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

- Consider solution operator $\mathcal{G}$

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

- p input, e.g. initial condition

Example

Poisson equation :

$\Delta u = f, \quad \mathcal{G} : f \mapsto u$

Time-dependent PDE :

$\mathcal{G} : u_0 \mapsto u_T$

Setting

- Even though we know the underlying PDE $\mathcal{D}u = f$
- finding the solution is complicated/costly/impractical
- the algorithm to approx. $\mathcal{G} : p \to u$ is complex to develop

Learn the solution operator from the data

Learn it automatically from data :

1. generate trainset from input/target pairs $(p_i, u_i)_{i=1,\dots,N}$ using solver,
2. train neural network $\mathcal{G}_\theta$ to minimize loss on trainset :

$$\frac{1}{N}\sum_{i}^{N} \|\mathcal{G}_\theta(p_i) - u_i\|^2$$

3. use $\mathcal{G}_\theta$ for new inputs p on the fly

Insights and differences with PINNs (and other PDE settings)

- Access to PDE only through the data pairs $(p_i, u_i)_{i=1,\dots,N}$
- can be applied when the PDE is unknown,
- less suffering from ill-conditioning, CFL, etc...
- no optimization for each input p : **much faster**
- **input is u, not x : generalization error becomes very important**

A prototypical example : the Fourier Neural Operator (FNO)

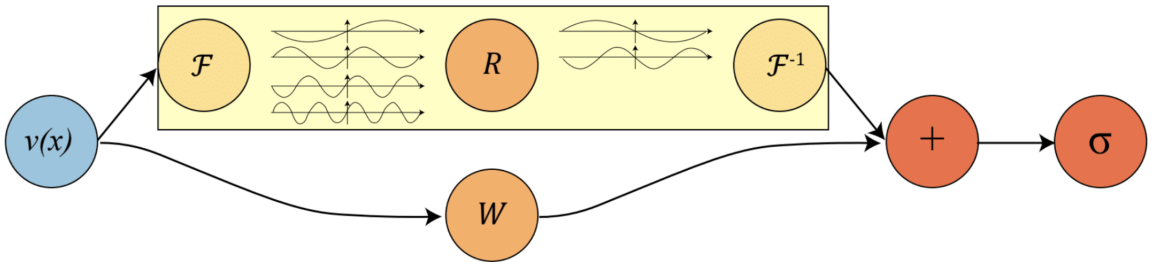


Figure 1 – layer of the Fourier neural operator

- neural operators received lots of attention for solving PDEs
- the *Fourier Neural Operator, 2021* is a prototypical example ($\sim$ 2500 citations),
- sequence of layers, as in classical nns

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

A prototypical example : the Fourier Neural Operator (FNO)

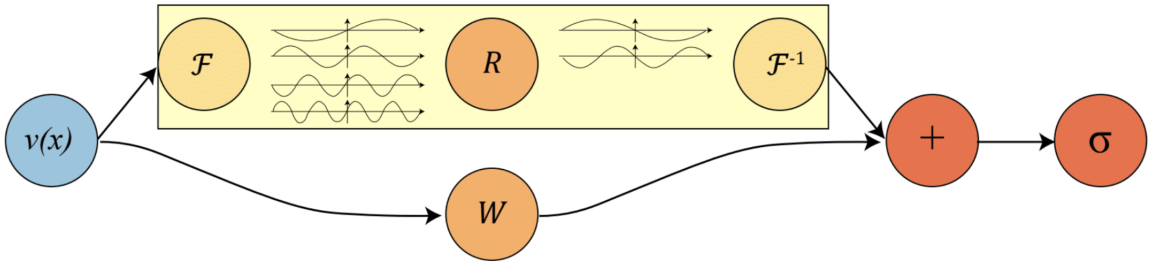


Figure 1 – layer of the Fourier neural operator

- neural operators received lots of attention for solving PDEs
- the *Fourier Neural Operator, 2021* is a prototypical example ($\sim$ 2500 citations),
- sequence of layers, as in classical nns

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

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

a prototypical example, the Fourier neural operator (FNO)

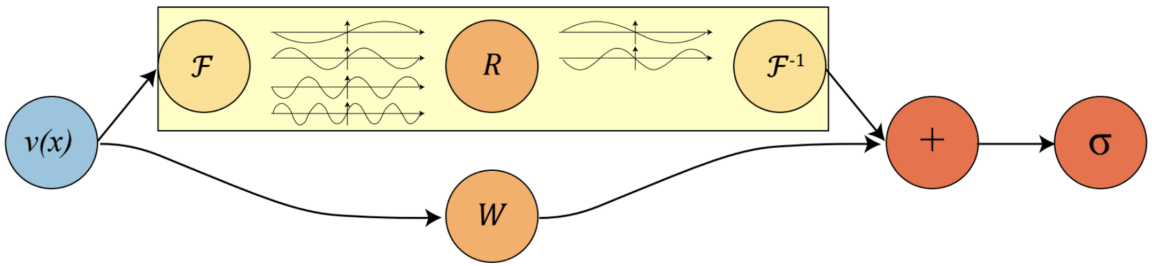


Figure 2 – layer of the Fourier neural operator

- for example, continuous convolution with Fourier layer :

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

a prototypical example, the Fourier neural operator (FNO)

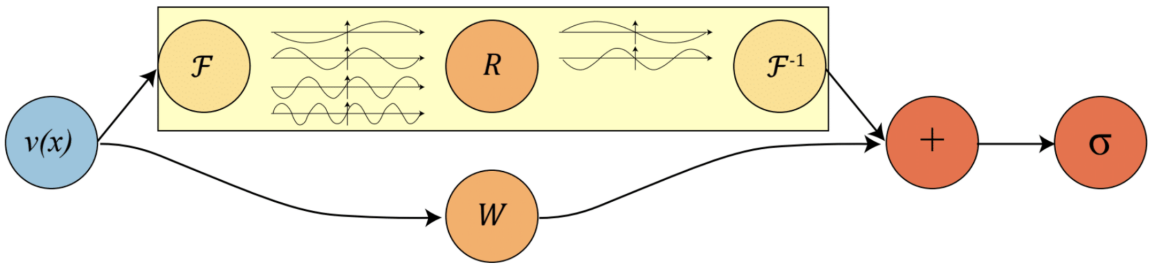


Figure 2 – layer of the Fourier neural operator

- for example, continuous convolution with Fourier layer :

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

- state of the art for many PDEs, claims to handle functions

a prototypical example, the Fourier neural operator (FNO)

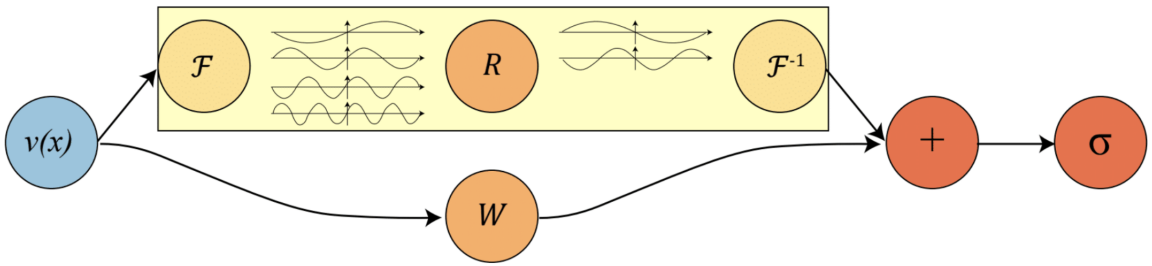


Figure 2 – layer of the Fourier neural operator

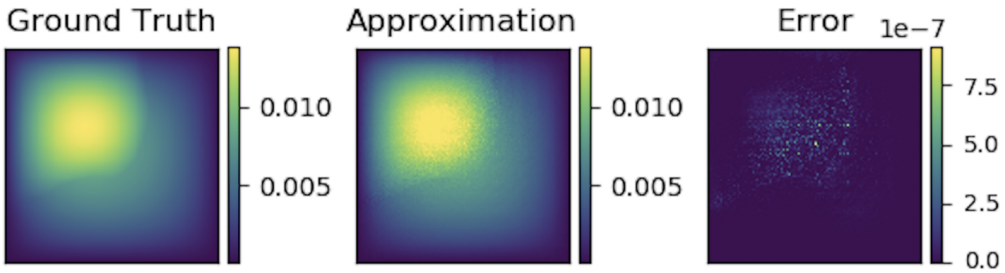
- for example, continuous convolution with Fourier layer :

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

- state of the art for many PDEs, claims to handle functions
- can easily go from one grid resolution to another, not CNNs (*zero-shot super resolution*),

Some results

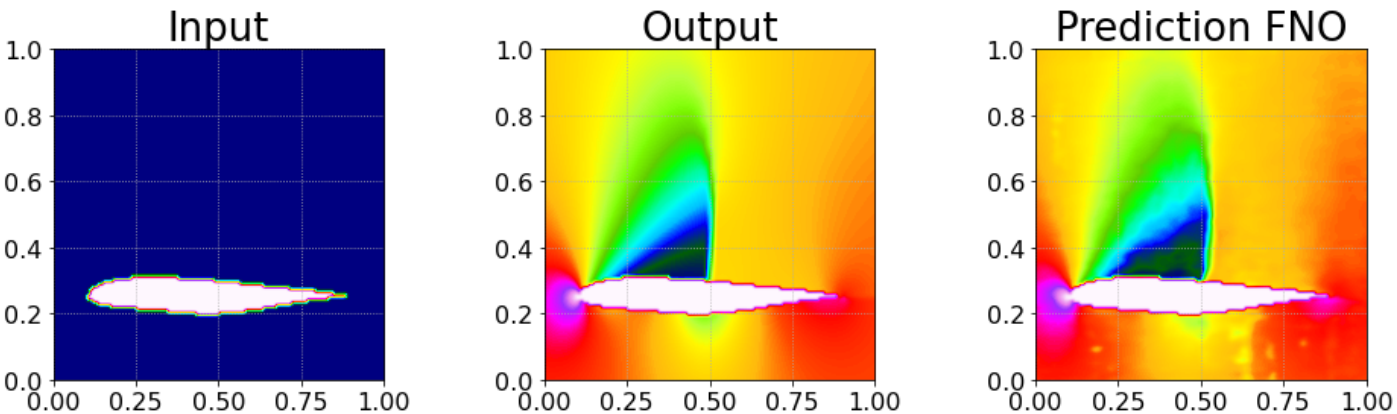
$$-\nabla \cdot (a(x)\nabla u(x)) = f(x) \qquad x \in (0,1)^2$$
$$u(x) = 0 \qquad x \in \partial(0,1)^2$$



Networks	$s = 85$	$s = 141$	$s = 211$	$s = 421$
NN	0.1716	0.1716	0.1716	0.1716
FCN	0.0253	0.0493	0.0727	0.1097
PCANN	0.0299	0.0298	0.0298	0.0299
RBM	0.0244	0.0251	0.0255	0.0259
DeepONet	0.0476	0.0479	0.0462	0.0487
GNO	0.0346	0.0332	0.0342	0.0369
LNO	0.0520	0.0461	0.0445	–
MGNO	0.0416	0.0428	0.0428	0.0420
FNO	**0.0108**	**0.0109**	**0.0109**	**0.0098**

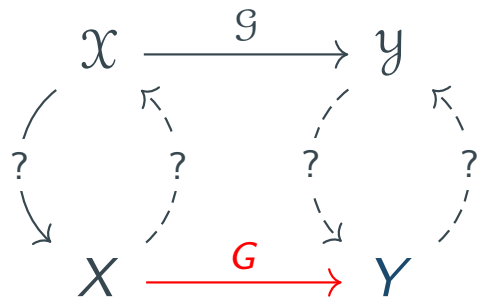
- Also SOTA for Navier-Stokes

What is happening here ?



Compressible Euler, flow passed airflow problem

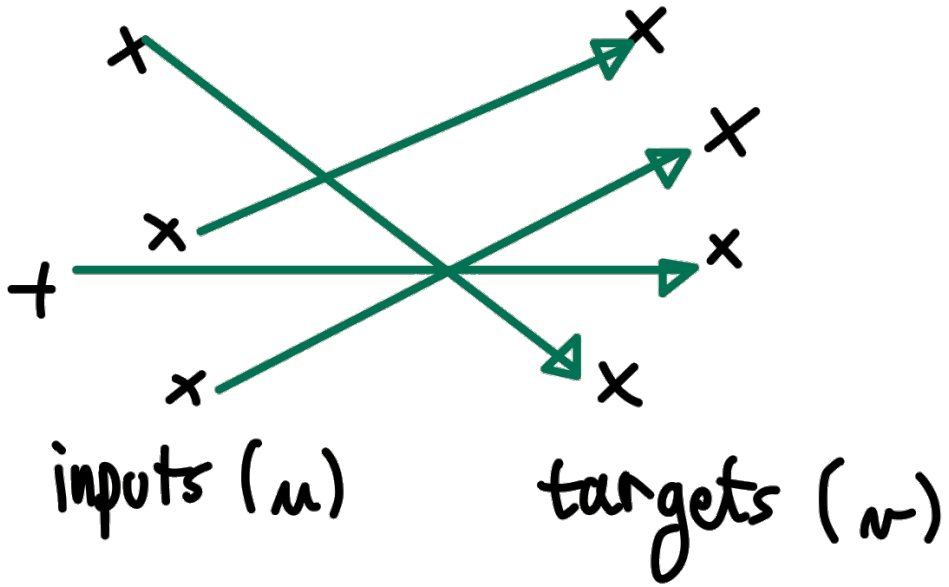
FNO in practice



- 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)

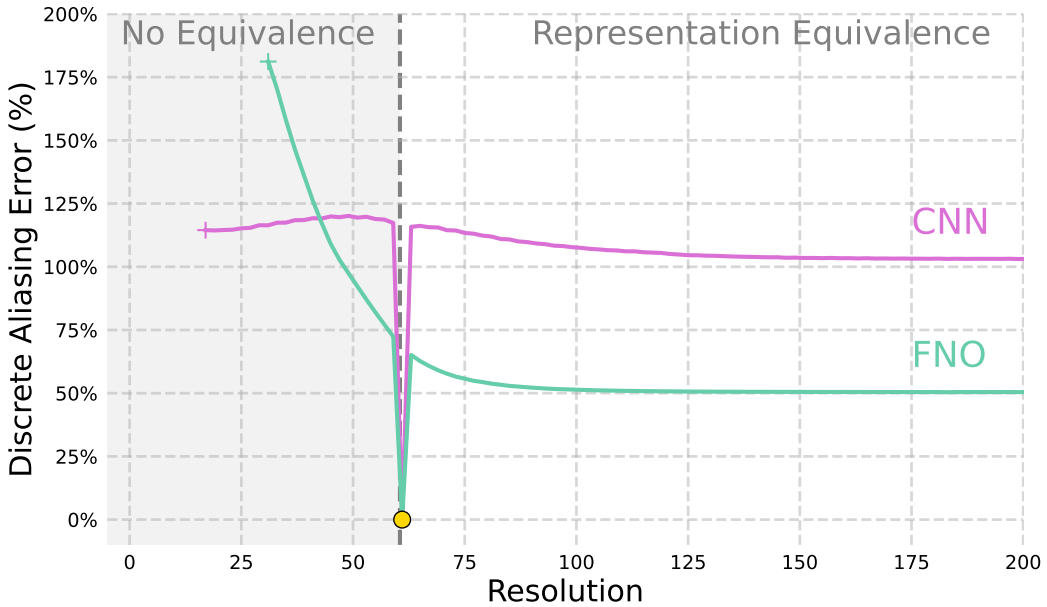
- link between operator $\mathcal{G}$ and discrete map G ?

simple experiment : learning a complex mapping



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

Discrete representations not equivalent

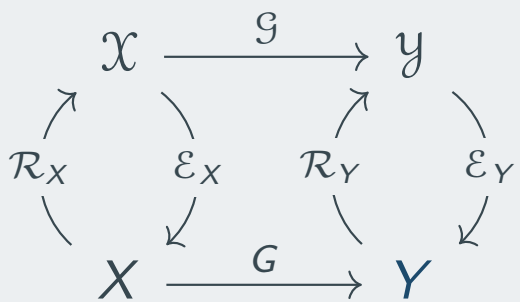


- a bit of an extreme case, but highlights
- discrepancy between continuous and discrete

Formalizing things

- *Representation equivalent neural operators : a framework for alias-free operator learning, FB, EdB, BR, RM, SM, RA* ***Neurips 2023***

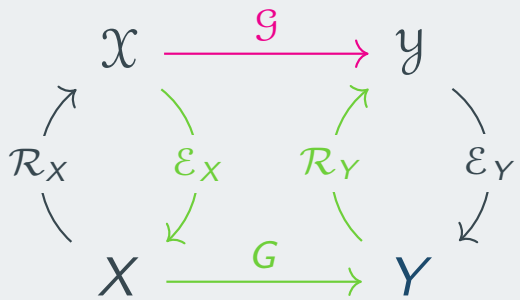
Quantifying continuous-discrete discrepancy



- $\mathcal{X}, \mathcal{Y}$ separable Hilbert spaces,
- $\mathcal{E}, \mathcal{R}$ encoding, reconstruction,
- ***Generalized aliasing error*** :

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

Quantifying continuous-discrete discrepancy

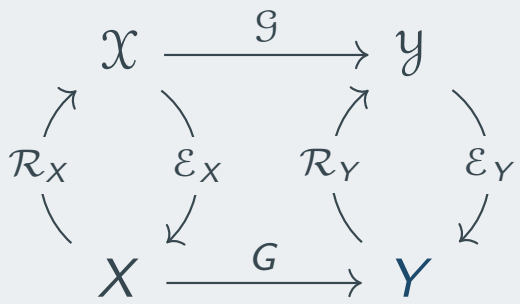


- $\mathcal{X}, \mathcal{Y}$ separable Hilbert spaces,
- $\mathcal{E}, \mathcal{R}$ encoding, reconstruction,
- ***Generalized aliasing error*** :

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

- if $\varepsilon(\mathcal{G}, G) = 0$, **diagram commutes** : $\mathcal{G} = \mathcal{R}_Y \circ G \circ \mathcal{E}_X$

Quantifying continuous-discrete discrepancy

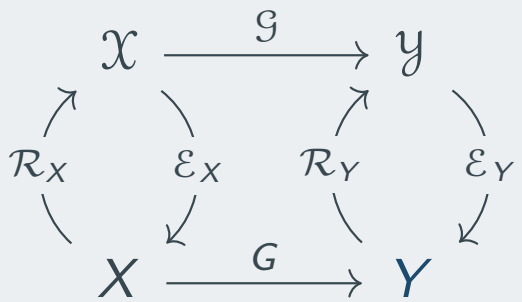


- $\mathcal{X}, \mathcal{Y}$ separable Hilbert spaces,
- $\mathcal{E}, \mathcal{R}$ encoding, reconstruction,
- ***Generalized aliasing error*** :

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

- if $\varepsilon(\mathcal{G}, G) = 0$, diagram commutes.
- **What about two discrete operators G and G' ?**

Quantifying continuous-discrete discrepancy



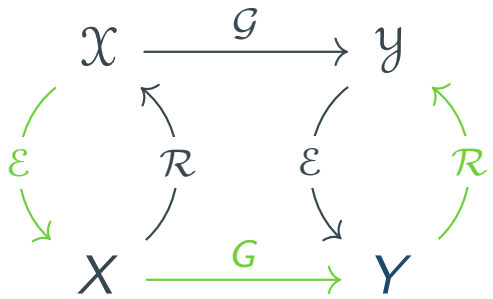
- $\mathcal{X}, \mathcal{Y}$ separable Hilbert spaces,
- $\mathcal{E}, \mathcal{R}$ encoding, reconstruction,
- ***Generalized aliasing error*** :

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

- if $\varepsilon(\mathcal{G}, G) = 0$, diagram commutes.
- 2 grids : G, G'. If aliasing is zero, **equivalence** :

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

- if $\varepsilon(G, \mathcal{G}) \neq 0$, potential non-equivalence for (G, G').
- this may be behind the FNO discrepancy

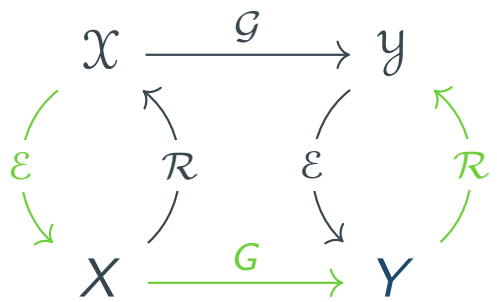


Instantiation : Bandlimited spaces

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

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

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

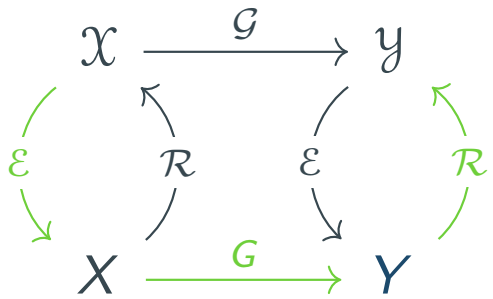


Instantiation : Bandlimited spaces

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

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

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



Instantiation : Bandlimited spaces

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

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

- 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.

Going back to FNO

per layer analysis

- continuous and discrete convolutions **equivalent**,

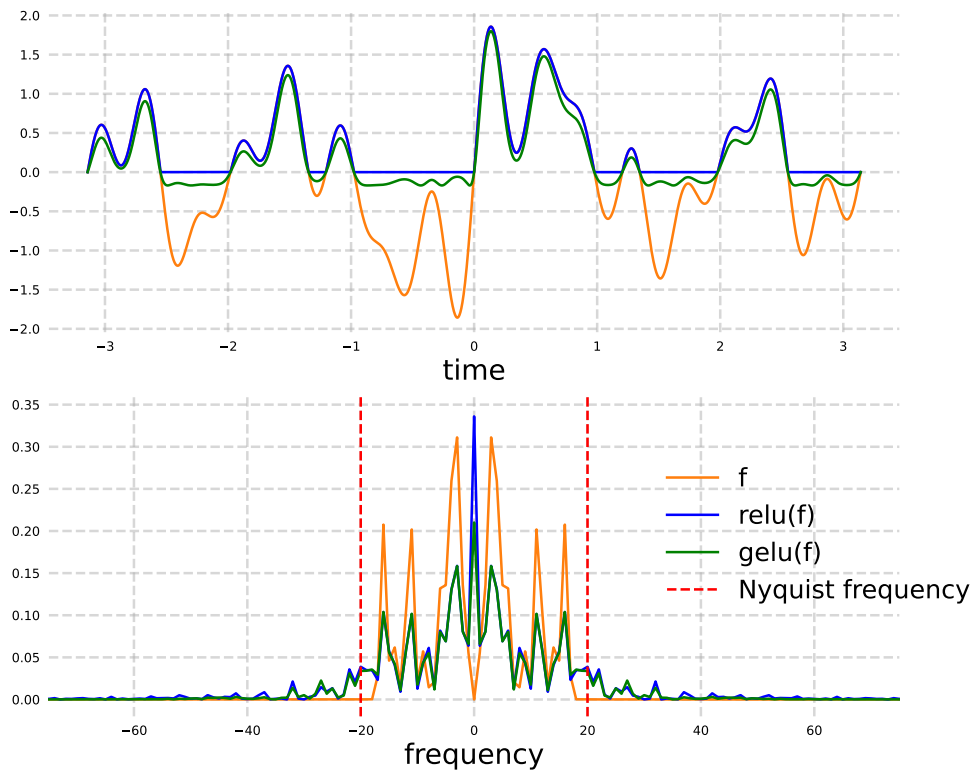
Going back to FNO

per layer analysis

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

per layer analysis

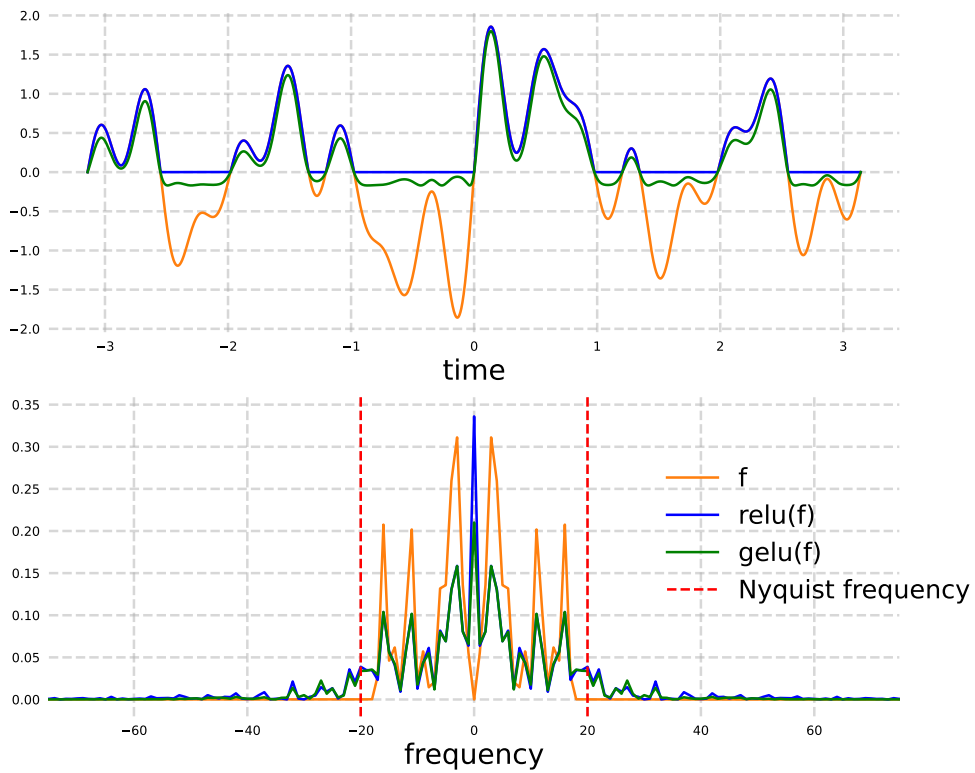
- continuous and discrete convolutions **equivalent**,
- activation function is not : $\varepsilon(G, \mathcal{G}) \neq 0$, i.e. diagram does not commute



per layer analysis

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

Is it fixable ?

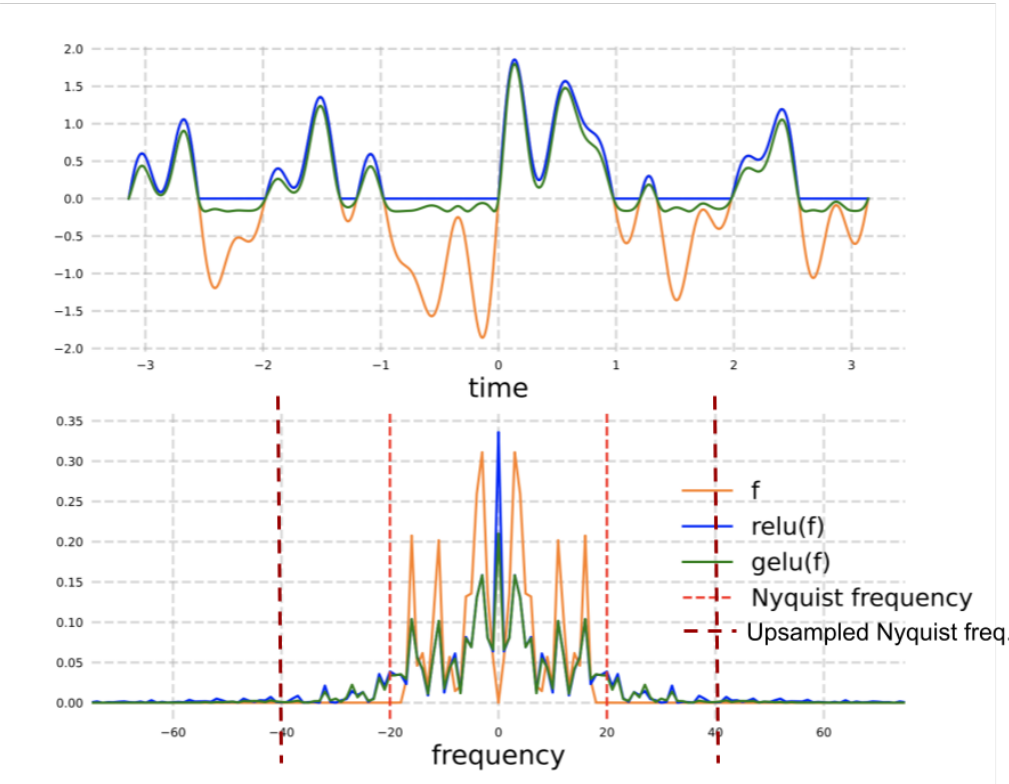


Fixing the activation function

$$\cancel{\Sigma(f) = \sigma(f)}, \qquad \Sigma(f) = [\mathcal{D} \circ \sigma \circ \mathcal{U}](f)$$

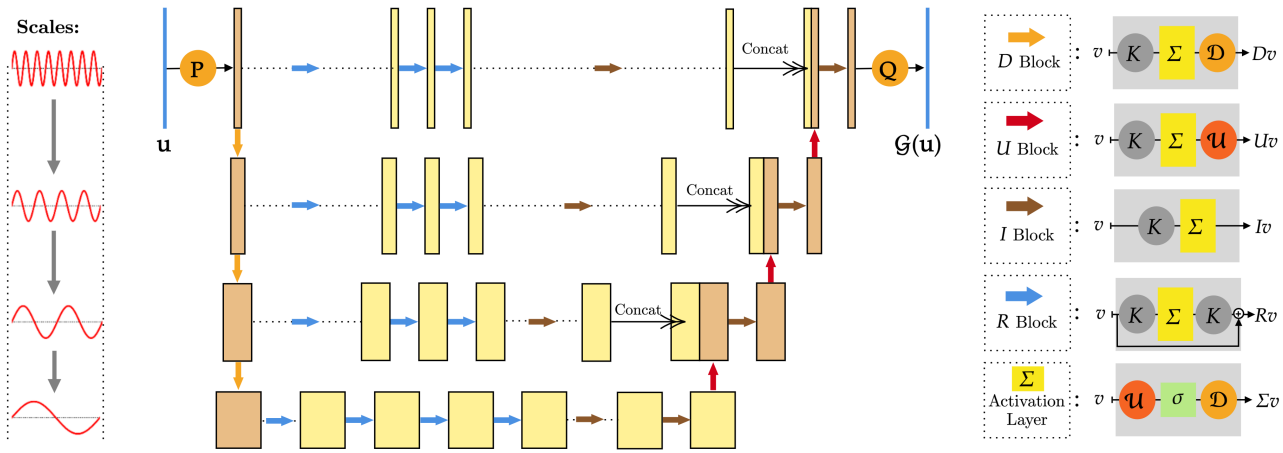
Just as in StyleGAN3, new activation :

- upsample
- activation
- downsample



Convolutional Neural Operators (CNO)

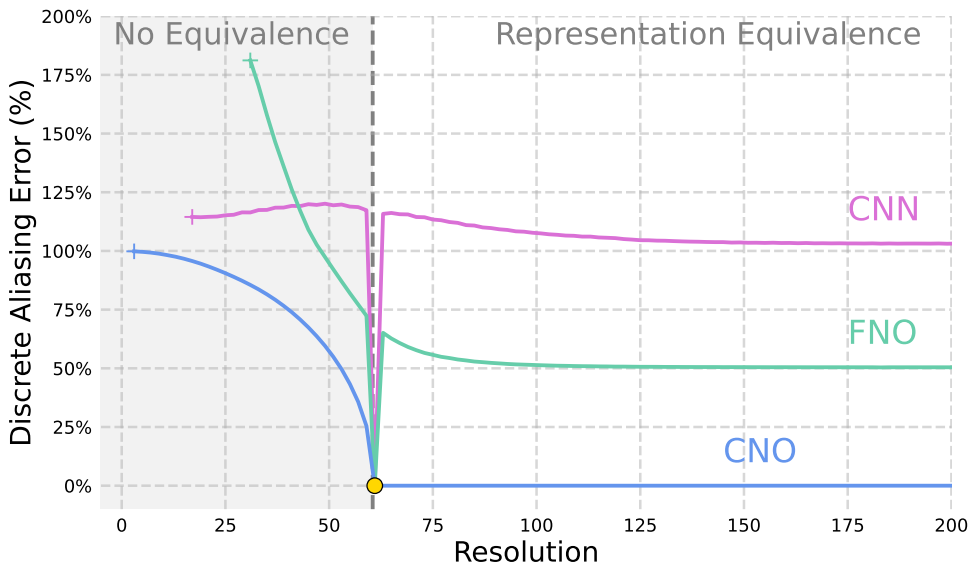
- ***Convolutional neural operators for robust and accurate learning of PDEs***, *B.R, R.M, FB, RA, SM, EdB,* ***Neurips 2023***



- Modified activation
- No need for Fourier Layers : Use convolutional layers,

results

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



quantitative results

Table 1 – Relative median L^1 test errors.

	FFNN	GT	UNet	ResNet	DON	FNO	CNO
Poisson	5.74%	2.77%	0.71%	0.43%	12.92%	4.98%	**0.21%**
Wave	2.51%	1.44%	1.51%	0.79%	2.26%	1.02%	**0.63%**
Smooth Transport	7.09%	0.98%	0.49%	0.39%	1.14%	0.28%	**0.24%**
Discontinuous Transport	13.0%	1.55%	1.31%	**1.01%**	5.78%	1.15%	**1.01%**
Allen-Cahn	18.27%	0.77%	0.82%	1.40%	13.63%	**0.28%**	0.54%
Navier-Stokes	8.05%	4.14%	3.54%	3.69%	11.64%	3.57%	**2.76%**
Darcy Flow	2.14%	0.86%	0.54%	0.42%	1.13%	0.80%	**0.38%**
Compressible Euler	0.78%	2.09%	0.38%	1.70%	1.93%	0.44%	**0.35%**

quantitative results

Table 1 – Relative median L^1 test errors.

	FFNN	GT	UNet	ResNet	DON	FNO	CNO
Poisson	5.74%	2.77%	0.71%	0.43%	12.92%	4.98%	**0.21%**
Wave	2.51%	1.44%	1.51%	0.79%	2.26%	1.02%	**0.63%**
Smooth Transport	7.09%	0.98%	0.49%	0.39%	1.14%	0.28%	**0.24%**
Discontinuous Transport	13.0%	1.55%	1.31%	**1.01%**	5.78%	1.15%	**1.01%**
Allen-Cahn	18.27%	0.77%	0.82%	1.40%	13.63%	**0.28%**	0.54%
Navier-Stokes	8.05%	4.14%	3.54%	3.69%	11.64%	3.57%	**2.76%**
Darcy Flow	2.14%	0.86%	0.54%	0.42%	1.13%	0.80%	**0.38%**
Compressible Euler	0.78%	2.09%	0.38%	1.70%	1.93%	0.44%	**0.35%**

- extensive grid search for all baselines,

quantitative results

Table 1 – Relative median L^1 test errors.

	FFNN	GT	UNet	ResNet	DON	FNO	CNO
Poisson	5.74%	2.77%	0.71%	0.43%	12.92%	4.98%	**0.21%**
Wave	2.51%	1.44%	1.51%	0.79%	2.26%	1.02%	**0.63%**
Smooth Transport	7.09%	0.98%	0.49%	0.39%	1.14%	0.28%	**0.24%**
Discontinuous Transport	13.0%	1.55%	1.31%	**1.01%**	5.78%	1.15%	**1.01%**
Allen-Cahn	18.27%	0.77%	0.82%	1.40%	13.63%	**0.28%**	0.54%
Navier-Stokes	8.05%	4.14%	3.54%	3.69%	11.64%	3.57%	**2.76%**
Darcy Flow	2.14%	0.86%	0.54%	0.42%	1.13%	0.80%	**0.38%**
Compressible Euler	0.78%	2.09%	0.38%	1.70%	1.93%	0.44%	**0.35%**

- extensive grid search for all baselines,
- e.g. for Navier Stokes, 10^{3-4} speedups wrt sota GPU codes

quantitative results

Table 1 – Relative median L^1 test errors.

	FFNN	GT	UNet	ResNet	DON	FNO	CNO
Poisson	5.74%	2.77%	0.71%	0.43%	12.92%	4.98%	**0.21%**
Wave	2.51%	1.44%	1.51%	0.79%	2.26%	1.02%	**0.63%**
Smooth Transport	7.09%	0.98%	0.49%	0.39%	1.14%	0.28%	**0.24%**
Discontinuous Transport	13.0%	1.55%	1.31%	**1.01%**	5.78%	1.15%	**1.01%**
Allen-Cahn	18.27%	0.77%	0.82%	1.40%	13.63%	**0.28%**	0.54%
Navier-Stokes	8.05%	4.14%	3.54%	3.69%	11.64%	3.57%	**2.76%**
Darcy Flow	2.14%	0.86%	0.54%	0.42%	1.13%	0.80%	**0.38%**
Compressible Euler	0.78%	2.09%	0.38%	1.70%	1.93%	0.44%	**0.35%**

- extensive grid search for all baselines,
- e.g. for Navier Stokes, 10^{3-4} speedups wrt sota GPU codes
- other convolutional architectures also very good across many PDEs

quantitative results

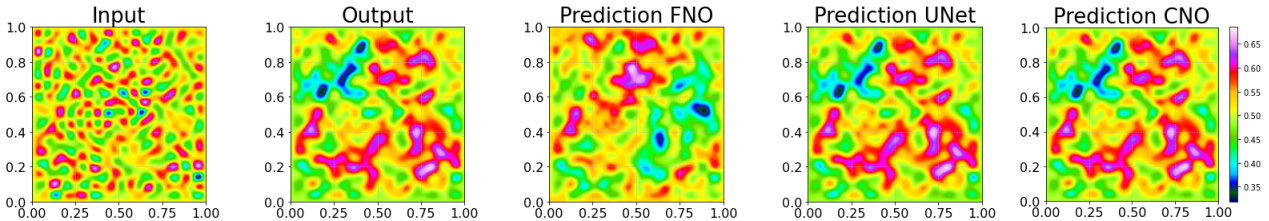
Table 1 – Relative median L^1 test errors.

	FFNN	GT	UNet	ResNet	DON	FNO	CNO
Poisson	5.74%	2.77%	0.71%	0.43%	12.92%	4.98%	**0.21%**
Wave	2.51%	1.44%	1.51%	0.79%	2.26%	1.02%	**0.63%**
Smooth Transport	7.09%	0.98%	0.49%	0.39%	1.14%	0.28%	**0.24%**
Discontinuous Transport	13.0%	1.55%	1.31%	**1.01%**	5.78%	1.15%	**1.01%**
Allen-Cahn	18.27%	0.77%	0.82%	1.40%	13.63%	**0.28%**	0.54%
Navier-Stokes	8.05%	4.14%	3.54%	3.69%	11.64%	3.57%	**2.76%**
Darcy Flow	2.14%	0.86%	0.54%	0.42%	1.13%	0.80%	**0.38%**
Compressible Euler	0.78%	2.09%	0.38%	1.70%	1.93%	0.44%	**0.35%**

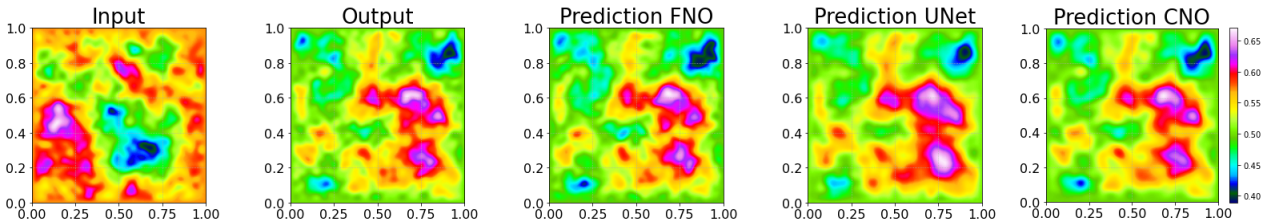
- extensive grid search for all baselines,
- e.g. for Navier Stokes, 10^{3-4} speedups wrt sota GPU codes
- other convolutional architectures also very good across many PDEs
- known to *generalize well*,

Qualitative Results

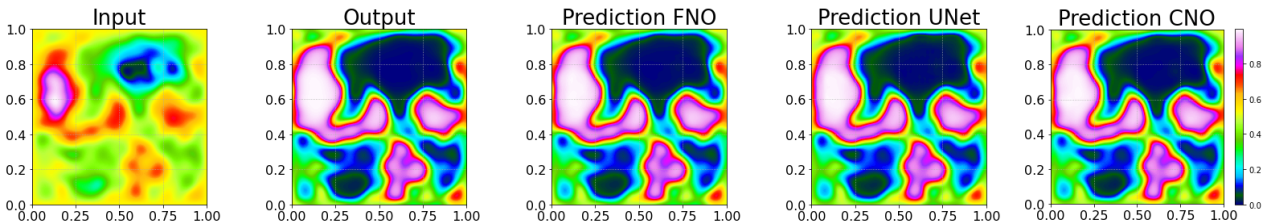
Poisson



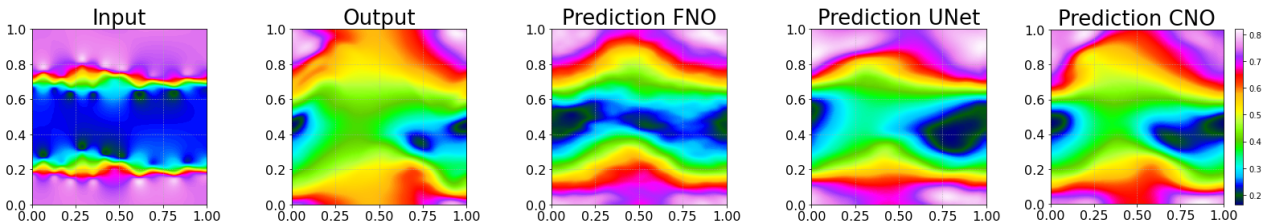
Wave



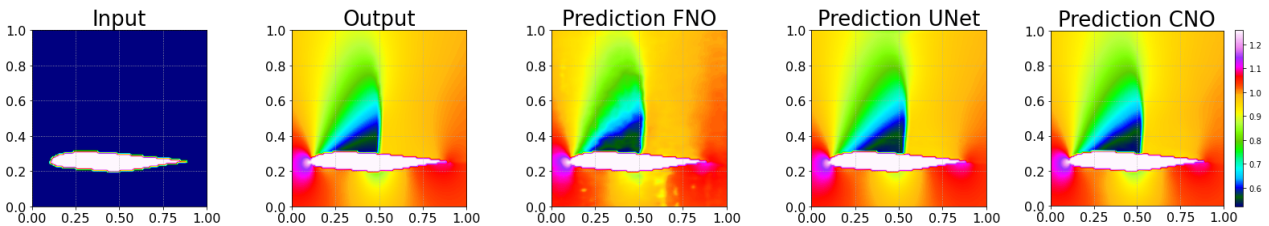
Allen-Cahn



Navier-Stokes



Compressible Euler



Neural Operators, Conclusion

- discrepancies between discrete and continuous operations in Neural Operators,

Neural Operators, Conclusion

- discrepancies between discrete and continuous operations in Neural Operators,
- as with any numerical method, we should pay attention to discretization errors,

Neural Operators, Conclusion

- discrepancies between discrete and continuous operations in Neural Operators,
- as with any numerical method, we should pay attention to discretization errors,
- **especially** important if training across grids or preserving continuous structures,

Neural Operators, Conclusion

- discrepancies between discrete and continuous operations in Neural Operators,
- as with any numerical method, we should pay attention to discretization errors,
- **especially** important if training across grids or preserving continuous structures,
- these errors can be assessed with tools from signal processing,

Neural Operators, Conclusion

- discrepancies between discrete and continuous operations in Neural Operators,
- as with any numerical method, we should pay attention to discretization errors,
- **especially** important if training across grids or preserving continuous structures,
- these errors can be assessed with tools from signal processing,
- discretization only one source of error : approx, training, **generalization** also very important

Taking a step back : Error Decomposition

$$\begin{aligned}\text{Total Error} \leq\ & \text{Approximation Error} \\ & + \text{Training Error} \\ & + \text{Generalization Gap}\end{aligned}$$

Taking a step back : Error Decomposition

$$\begin{aligned}\text{Total Error} \leq\ & \text{Approximation Error} \\ & + \text{Training Error} \\ & + \text{Generalization Gap}\end{aligned}$$

- Error made by the solver that generated the data, indep. from NN,
  the NN learns the solver's mistakes

Taking a step back : Error Decomposition

$$\begin{aligned}\text{Total Error} \leq\ & \text{Approximation Error} \\ & + \text{Training Error} \\ & + \text{Generalization Gap}\end{aligned}$$

- Error made by the NN on the training set, usually close to zero,

Taking a step back : Error Decomposition

$$\begin{aligned}\text{Total Error} \leq\ & \text{Approximation Error} \\ & + \text{Training Error} \\ & + \text{Generalization Gap}\end{aligned}$$

- This measures performance on unseen data. Leading error

Taking a step back : Error Decomposition

$$\begin{aligned}\text{Total Error} \leq\ & \text{Approximation Error} \\ & + \text{Training Error} \\ & + \text{Generalization Gap}\end{aligned}$$

- This measures performance on unseen data. Leading error

Generalization Gap

$$\text{Generalization Gap} = h\,(\text{Model}, \text{Data})$$

$$\leq O\left(\sqrt{\frac{\text{proxy for model class size}}{\text{dataset size}}}\right)$$

Very little is known, conventional wisdom :

- Generalization error increases with model class size
  $\rightarrow$ more prior information $\Rightarrow$ better generalization

Generalization Gap

$$\text{Generalization Gap} = h\,(\text{Model}, \text{Data})$$
$$\leq O\left(\sqrt{\frac{\text{proxy for model class size}}{\text{dataset size}}}\right)$$

Very little is known, conventional wisdom :

- Generalization error increases with model class size
  $\rightarrow$ more prior information $\Rightarrow$ better generalization
- Generalization error decreases with dataset size,
  $\rightarrow$ more data $\Rightarrow$ better generalization

And

- bounds exist, as a function of network width, depth, weight norms
- bounds are indep. of data,
- hyperparam. Selection w/ validation set not modelled,

Generalization can be counterintuitive

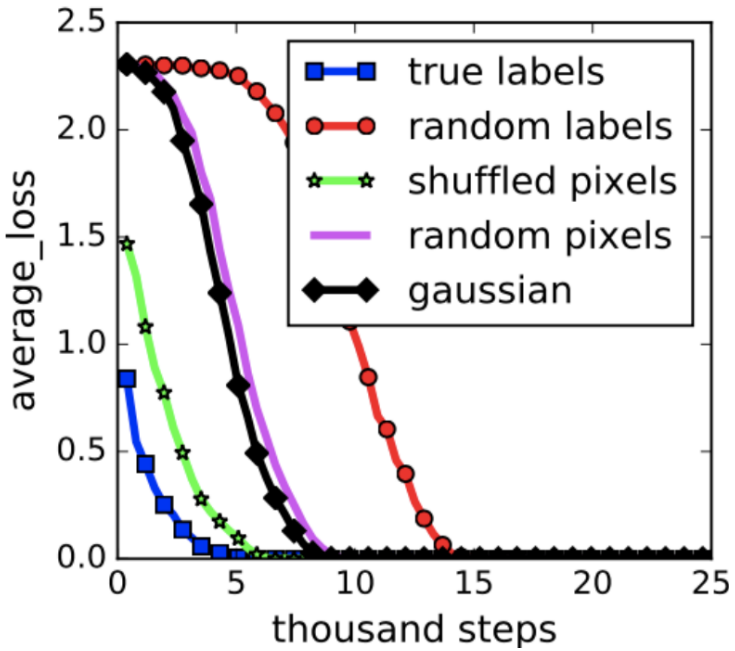


Figure 3 – Deep learning requires rethinking generalization, 2017

- Training loss zero,

Generalization can be counterintuitive

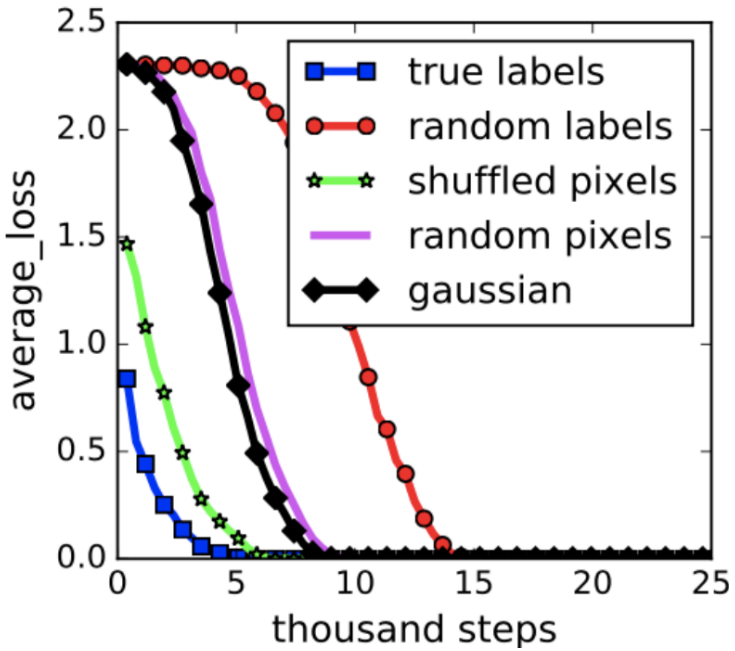


Figure 3 – Deep learning requires rethinking generalization, 2017

- Training loss zero,
- model class size is huge,

Generalization can be counterintuitive

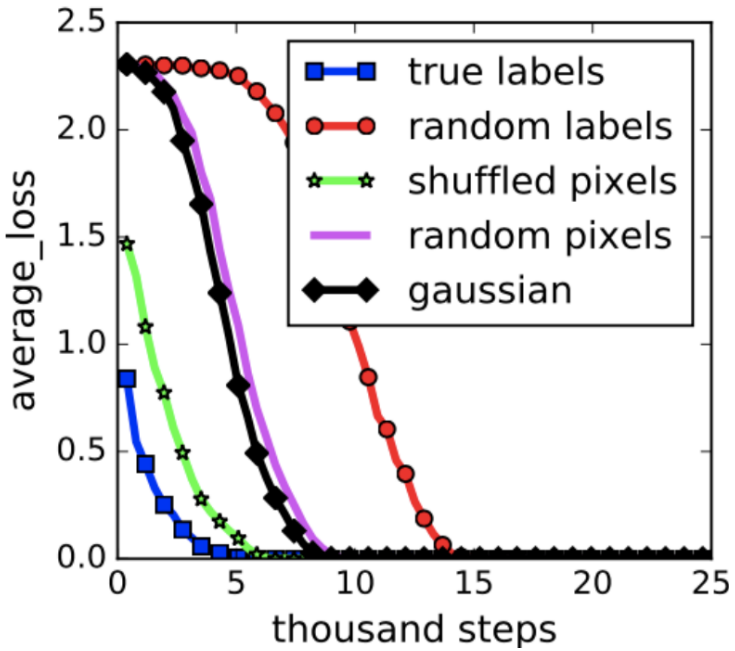


Figure 3 – Deep learning requires rethinking generalization, 2017

- Training loss zero,
- model class size is huge,
- classical (uniform) bounds tend to be vacuous,

Generalization can be counterintuitive

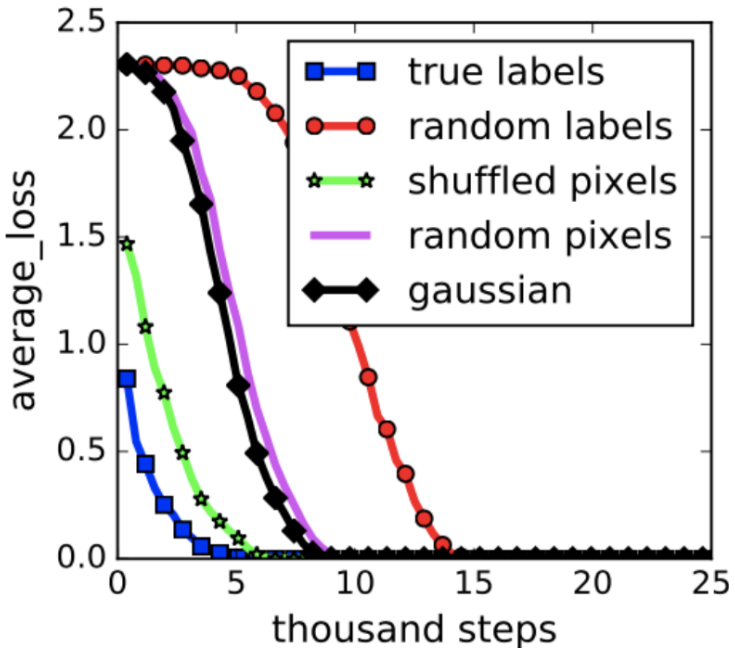
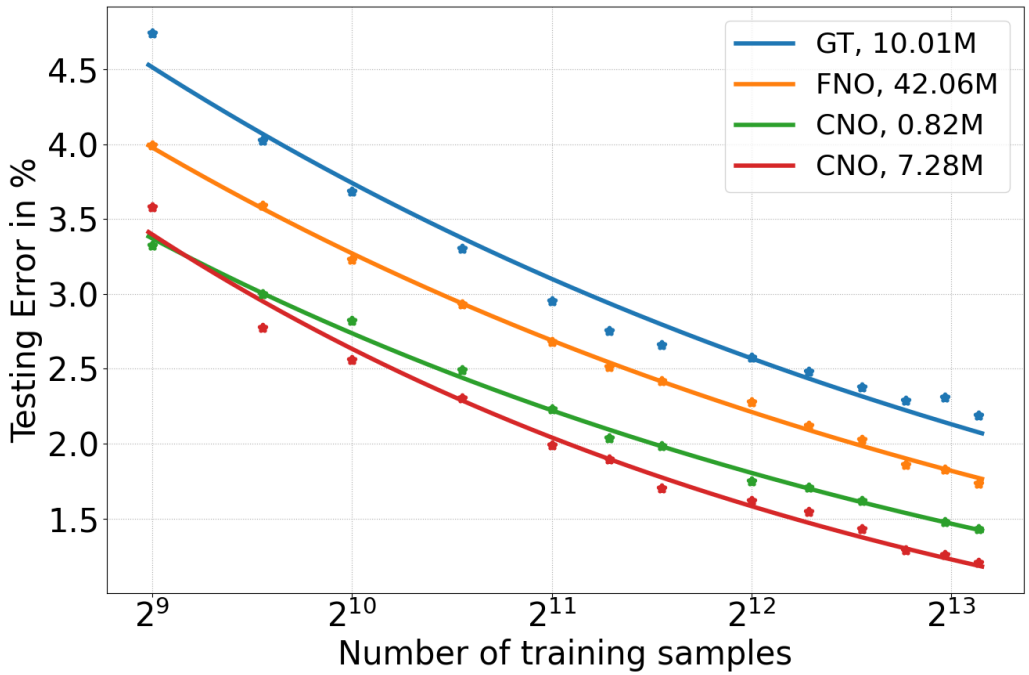


Figure 3 – Deep learning requires rethinking generalization, 2017

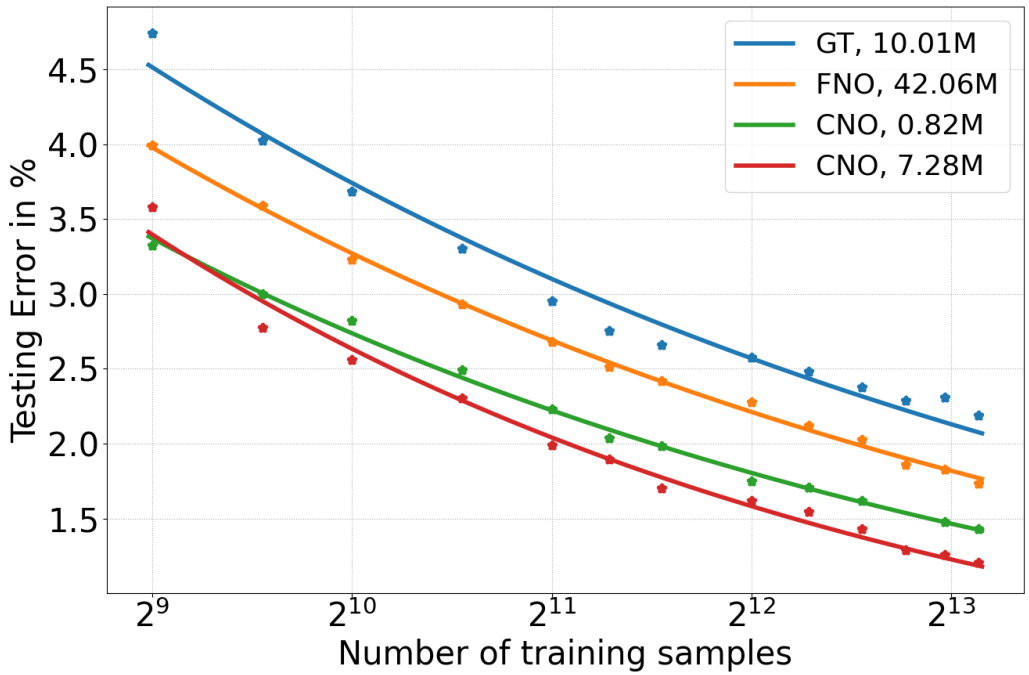
- Training loss zero,
- model class size is huge,
- classical (uniform) bounds tend to be vacuous,
- generalization ultimately depends on the data,

Dependance on dataset size ?



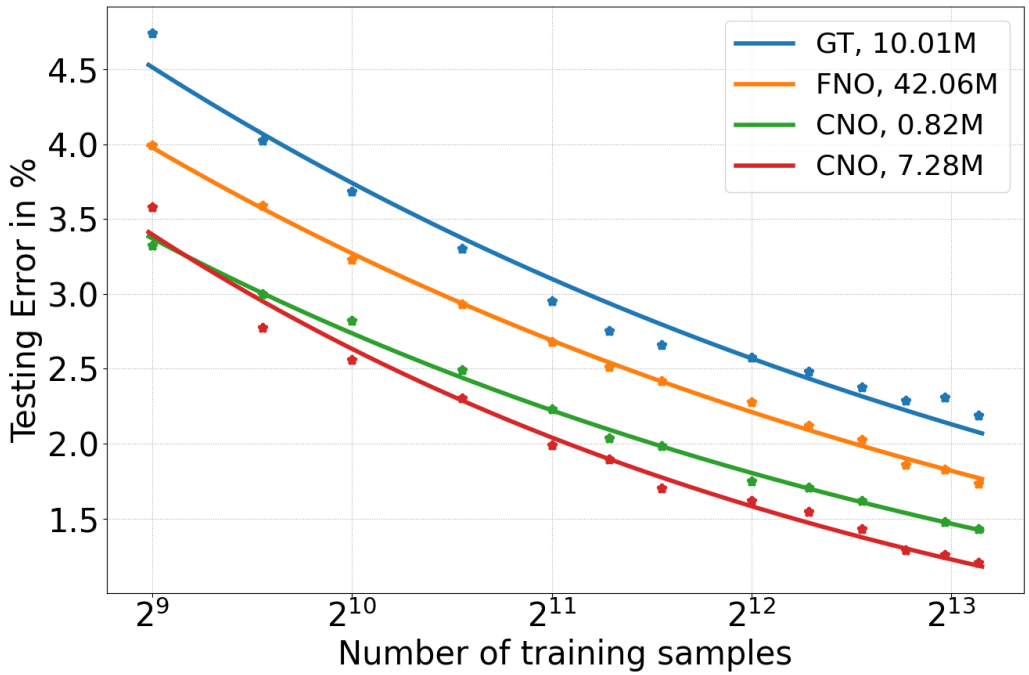
- we have seen class size is hard to model mathematically,

Dependance on dataset size ?



- we have seen class size is hard to model mathematically,
- dependence on data size seems clearer

Dependance on dataset size ?



- we have seen class size is hard to model mathematically,
- dependence on data size seems clearer
- Often predictable **scaling laws** when data drawn from fixed distr,

Idea : Focus on the data, regroup

- ***POSEIDON : Efficient Foundation Models for PDEs***, *M.H., B.R., T.R., R.K., R.M., EdB, S.M.* ***Neurips 2024***

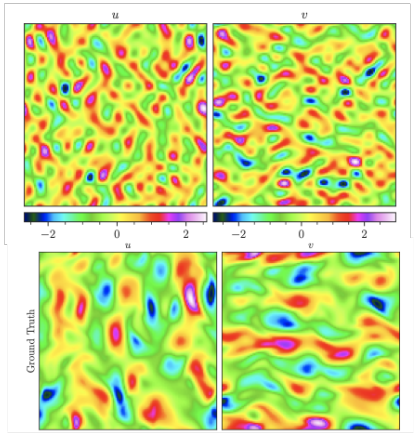


Figure 55: NS-Sines. Visualization of a random sample.

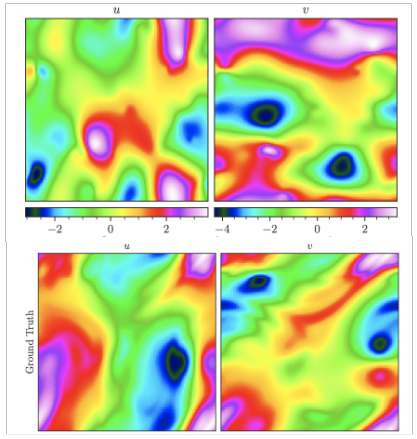


Figure 56: NS-Gauss. Visualization of a random sample.

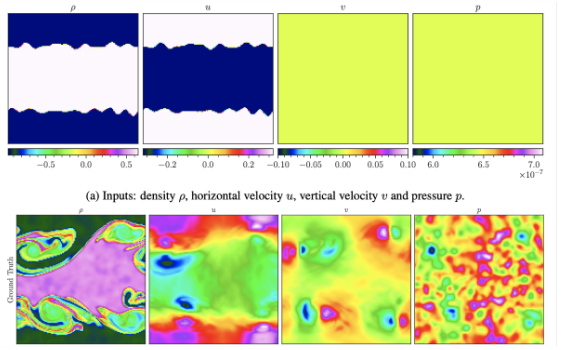


Figure 58: CE-CRP. Visualization of a random sample.

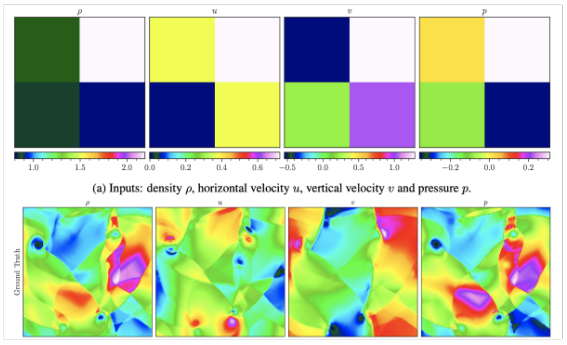


Figure 59: CE-KH. Visualization of a random sample.

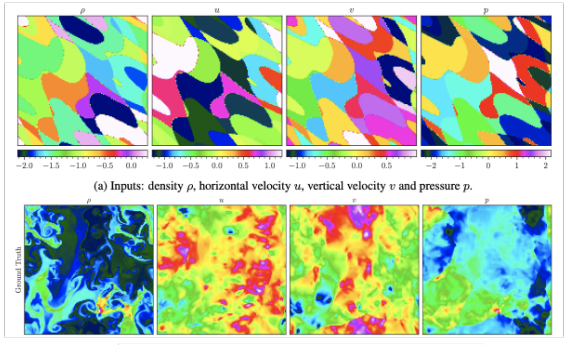


Figure 57: CE-RP. Visualization of a random sample.

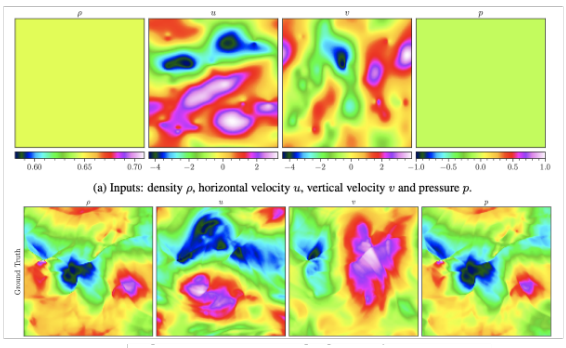
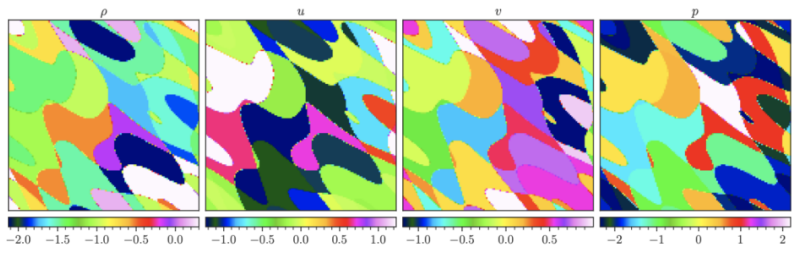


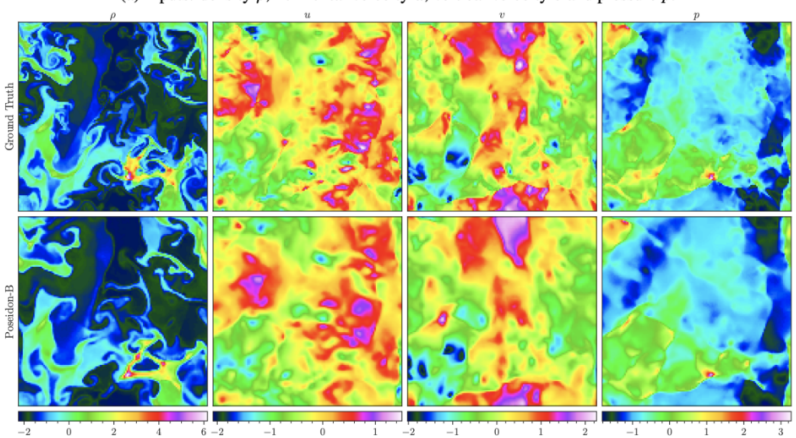
Figure 60: CE-Gauss. Visualization of a random sample.

- We consider a **larger, more diverse training dataset**,
- different equations : Navier-Stokes and Compressible Euler

Results

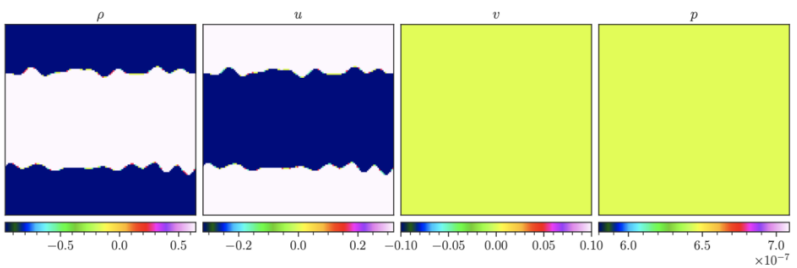


(a) Inputs: density ρ, horizontal velocity u, vertical velocity v and pressure p.

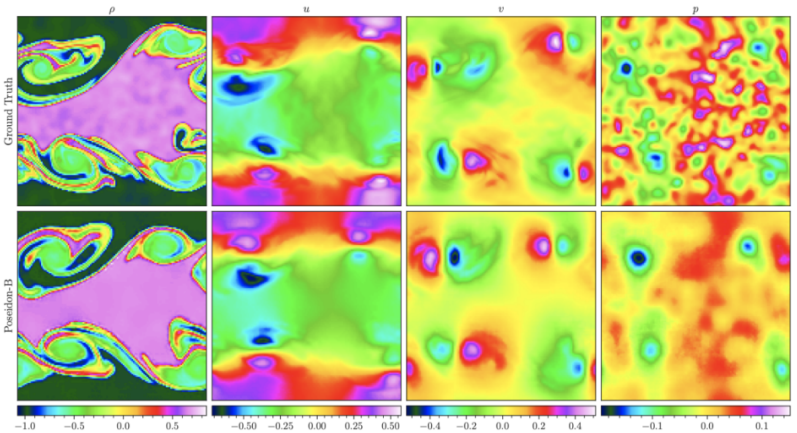


(b) (Top) Ground truth. (Bottom) Samples predicted by POSEIDON-B at $T = 1$.

Figure 58: CE-CRP. Visualization of a random sample.



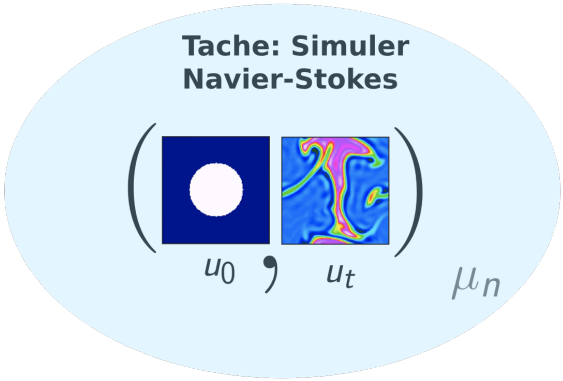
(a) Inputs: density ρ, horizontal velocity u, vertical velocity v and pressure p.

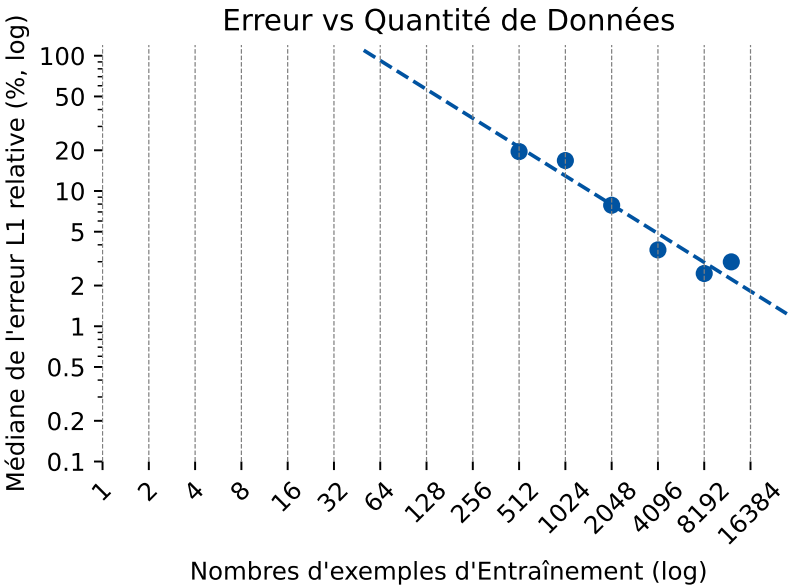


(b) (Top) Ground truth. (Bottom) Samples predicted by POSEIDON-B at $T = 1$.

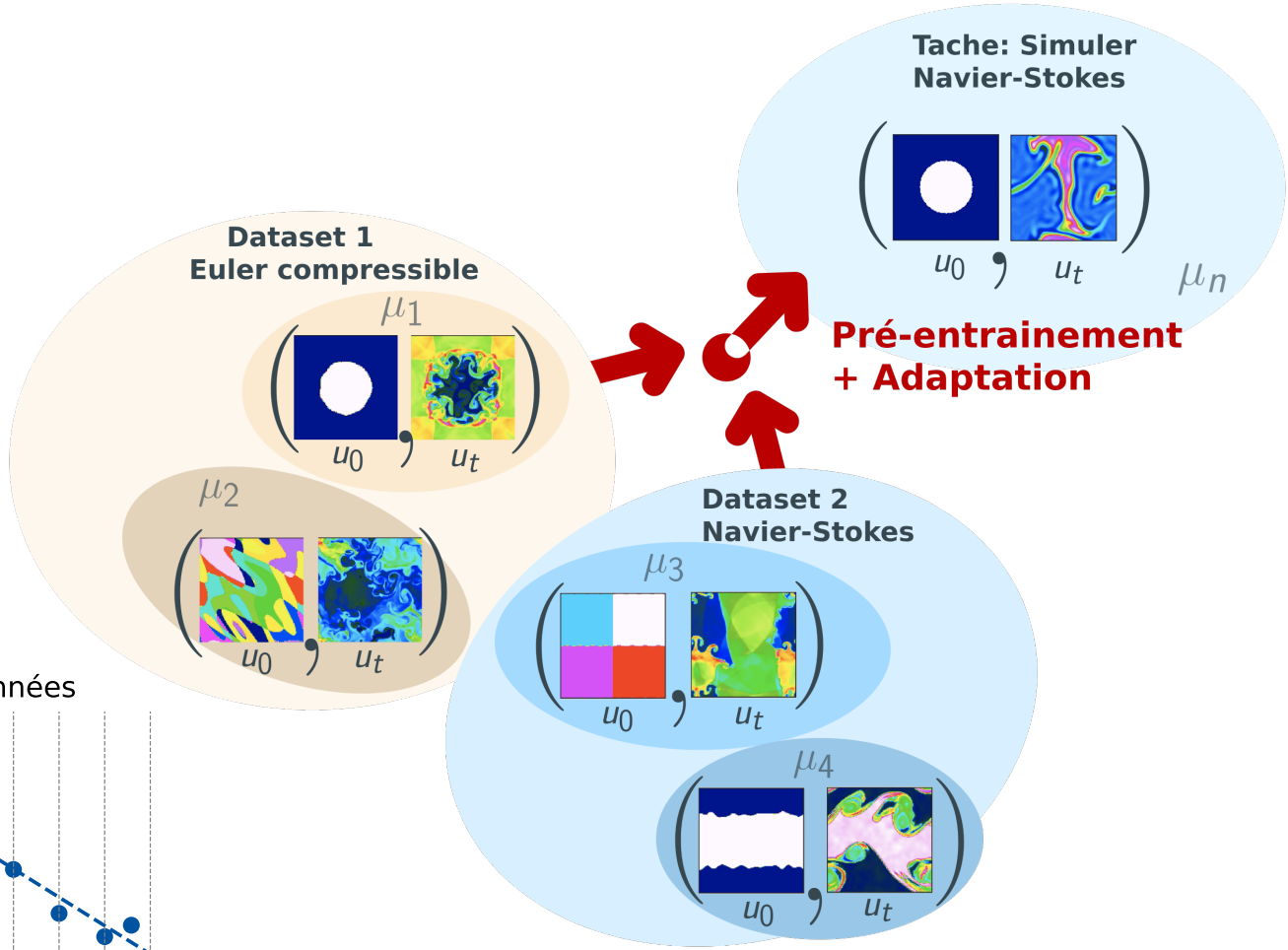
Figure 59: CE-KH. Visualization of a random sample.

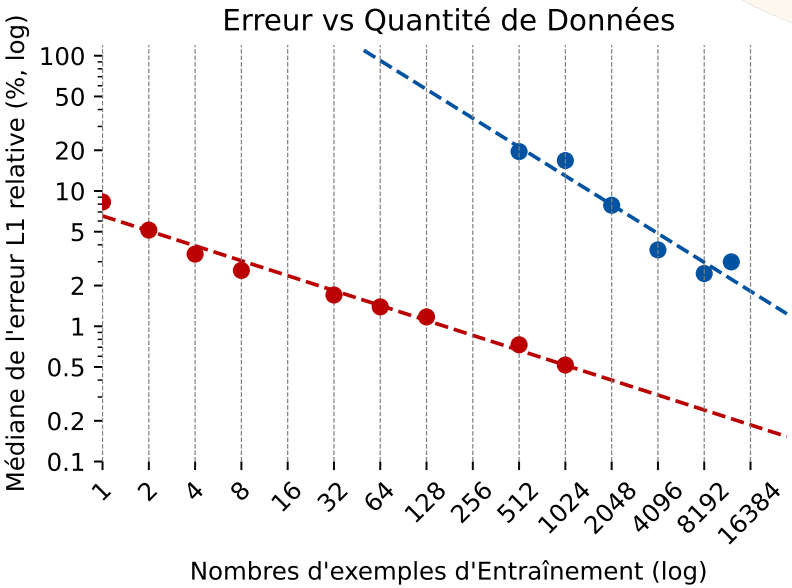
Given a new task, fine-tune





Given a new task, fine-tune





Quantative Results

	Pretrained Models						Models trained from Scratch					
	POSEIDON-L		CNO-FM		MPP-B		CNO		scOT		FNO	
	EG	*AG*	EG	*AG*	EG	*AG*	EG	*AG*	EG	*AG*	EG	*AG*
NS-PwC	**890.6**	***24.7***	16.6	*3.3*	7.4	*2.3*	3.7	*1.5*	5.4	*2.0*	1	*1*
NS-SVS	**502.9**	***7.3***	59.6	*3.1*	34.8	*2.2*	73.2	*3.4*	10.2	*1.2*	1	*1*
NS-BB	**552.5**	***29.3***	10.6	*3.9*	4.6	*2.6*	2.7	*1.7*	3.4	*2.1*	1	*1*
NS-SL	**21.9**	***5.5***	0.4	*0.8*	0.3	*0.8*	0.8	*1.2*	0.3	*0.8*	1	*1*
NS-Tracer-PwC	**49.8**	***8.7***	17.8	*3.6*	8.5	*2.7*	4.6	*1.9*	4.6	*1.9*	1	*1*
FNS-KF	**62.5**	***7.4***	13.2	*2.7*	2.0	*1.6*	3.1	*1.5*	3.3	*0.9*	1	*1*
CE-RPUI	**352.2**	***6.5***	33.2	*2.3*	0.0	*1.2*	12.5	*1.8*	15.6	*2.1*	1	*1*
CE-RM	**4.6**	***1.2***	0.6	*1.0*	0.0	*0.2*	1.7	*1.1*	0.4	*1.0*	1	*1*
SE-AF	3.4	*1.2*	4.8	*1.3*	2.2	*1.1*	**5.5**	***1.5***	1.2	*1.0*	1	*1*
GCE-RT	**5.3**	***2.0***	1.2	*1.0*	0.0	*0.3*	1.2	*1.4*	1.1	*1.1*	1	*1*
Wave-Layer	**46.5**	***6.1***	5.6	*2.2*	0.0	*0.9*	11.4	*3.0*	13.0	*2.9*	1	*1*
Wave-Gauss	**62.1**	***5.6***	6.0	*1.8*	0.0	*0.8*	14.0	*2.6*	9.2	*2.1*	1	*1*
ACE	**17.0**	***11.6***	1.7	*2.0*	0.0	*0.3*	4.5	*4.6*	6.5	*5.2*	1	*1*
Poisson-Gauss	**42.5**	***20.5***	25.0	*9.2*	17.0	*7.3*	21.1	*7.0*	9.8	*5.3*	1	*1*
Helmholtz	**78.3**	*6.1*	54.0	*5.1*	22.4	*3.0*	68.9	*7.3*	60.4	***9.0***	1	*1*

- **EG** : Efficiency Gain, ratio of no samples needed to reach same acc. as FNO on 1024 samples,
- **AG** : Accuracy Gain, wrt FNO

- rather generic model based on Swin Transformer,

Short conclusion

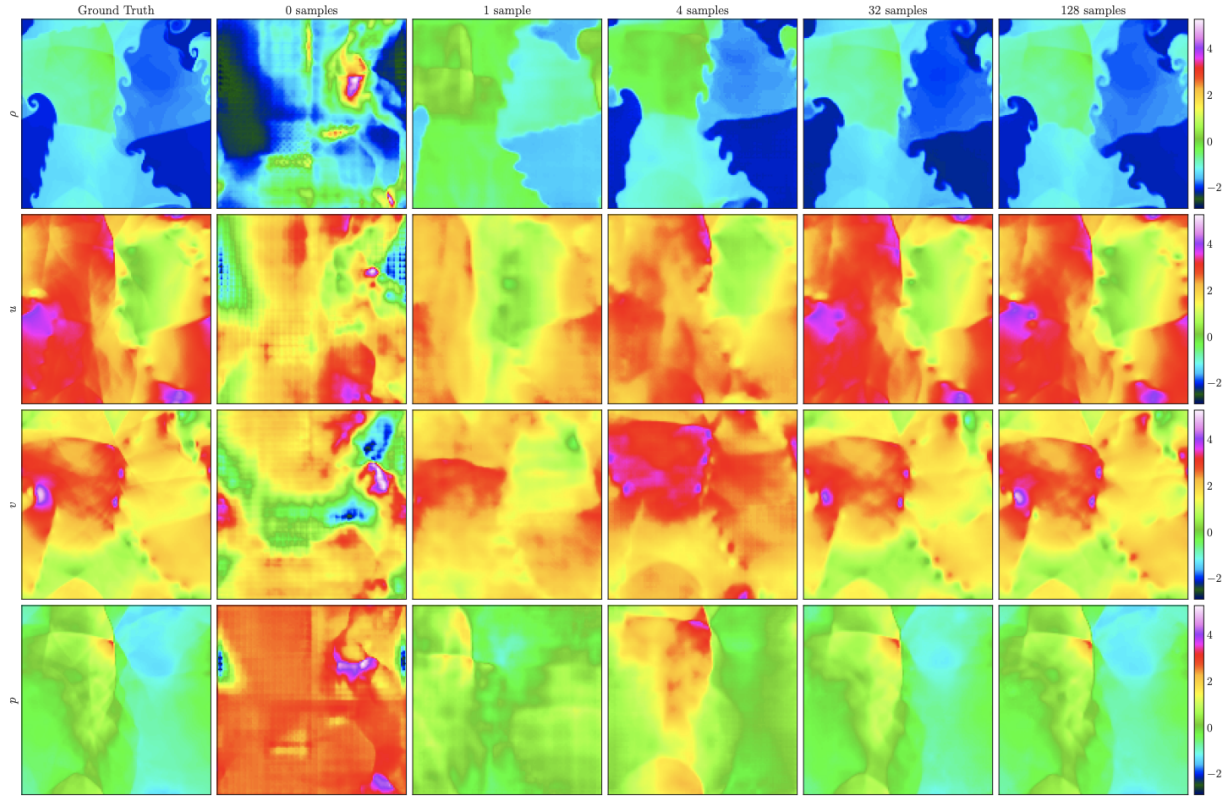


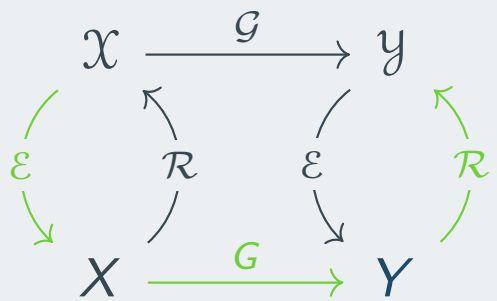
Figure 39: How POSEIDON-B approximates a random sample for the CE-RPUI task when trained with different numbers of task-specific trajectories.

- Neural operators are a promising tool for solving PDEs,
- Generalisation error is key, and a focus on data is important

Appendix

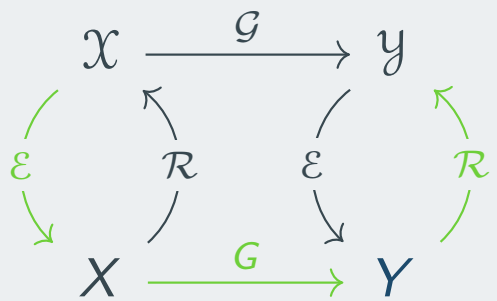
Representation equivalent neural operators (ReNO)

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



Representation equivalent neural operators (ReNO)

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

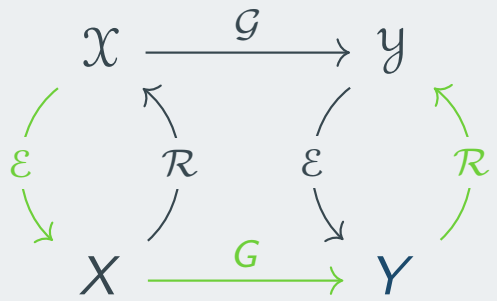


- i.e. discrepancy between continuous and discrete, aka *aliasing*

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

Representation equivalent neural operators (ReNO)

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



- 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

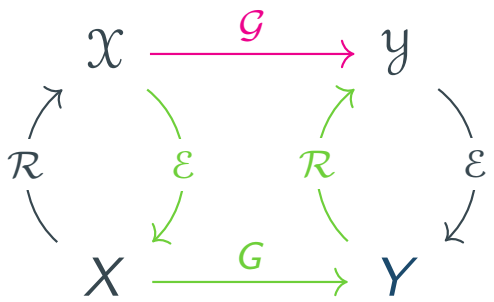
- Design each layer $(\mathcal{G}_\ell, G_\ell)$ such that diagram **commutes** :

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

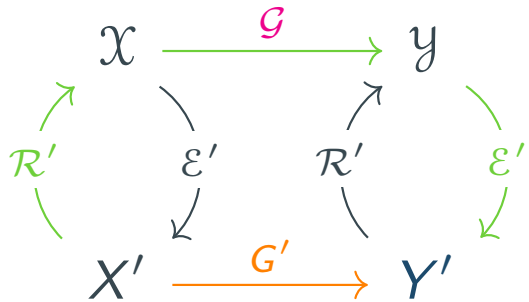
- composition of layers is also a ReNO

Equivalence between discrete representations

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

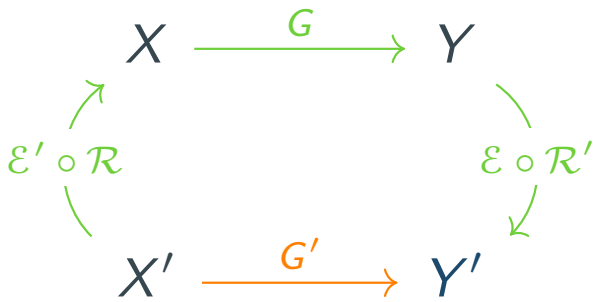


$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E}$$



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

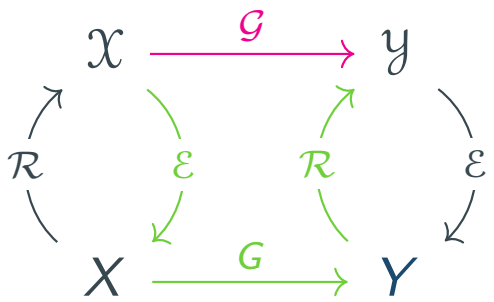
If both diagrams commute, **discrete representations are equivalent** :



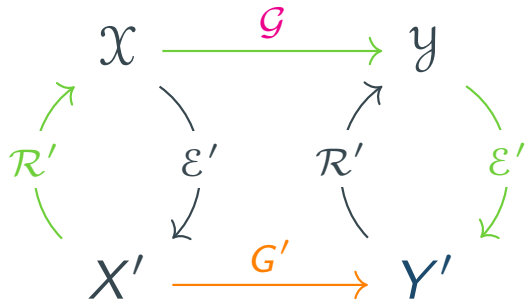
$$G' = \mathcal{E}' \circ \mathcal{R} \circ G \circ \mathcal{E} \circ \mathcal{R}'$$

Equivalence between discrete representations

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

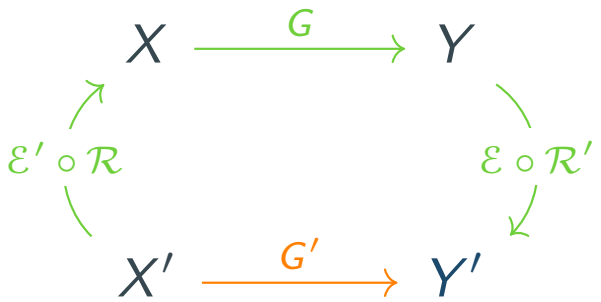


$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E}$$



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

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

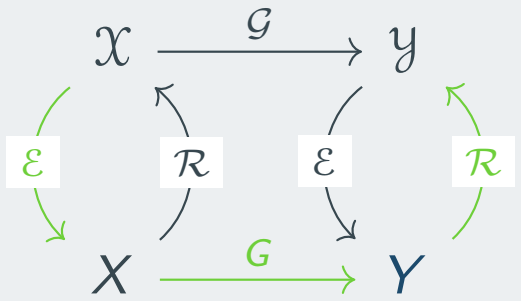


$$G' = \mathcal{E}' \circ \mathcal{R} \circ G \circ \mathcal{E} \circ \mathcal{R}'$$

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

Representation equivalent neural operators (ReNO)

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

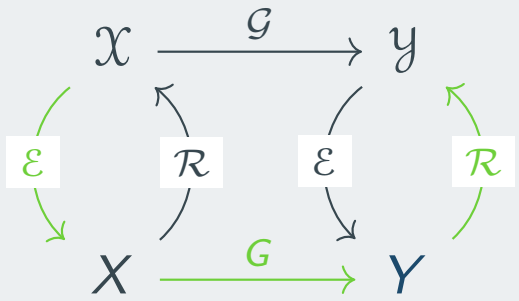


- $f = \sum_{i \in I} c_i \Phi_i(x)$. discretize : $f \xrightarrow{\mathcal{E}} c$, reconstruct : $c \xrightarrow{\mathcal{R}} f$
- $\{\Phi_i\}_{i \in I}$ basis or frame spanning $\mathcal{X}, \mathcal{Y}$,
- i.e. discrepancy between continuous and discrete, aka *aliasing*

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

Representation equivalent neural operators (ReNO)

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



- $f = \sum_{i \in I} c_i \Phi_i(x)$. 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...

synthesis and analysis operators

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

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

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

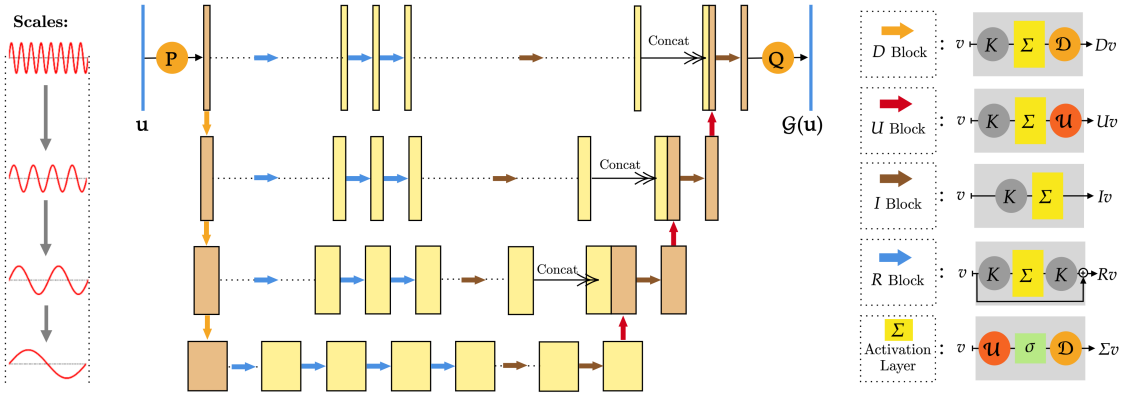
where the series converge unconditionally.

Back to classical convolutions

- classical convs have very good performance in classical ML,
- convs in Fourier not required, classical ones work too :

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

CNO : Unet-like architecture



- upsampling $\mathcal{U}$ and downsampling $\mathcal{D}$: interpolation with sinc filter
- adding residual connections in the skip connections helps,
- all layers are (approximate) ReNOs, in the sense

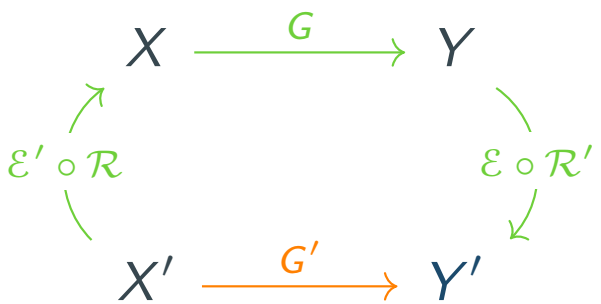
$$\mathcal{G}_\ell = \mathcal{R} \circ G_\ell \circ \mathcal{E}$$

- simple way to go from one resolution to another :

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

from one resolution to another

If each layer's diagram commutes, just use the smallest resolution and interpolate back :



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

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%**

Additional details

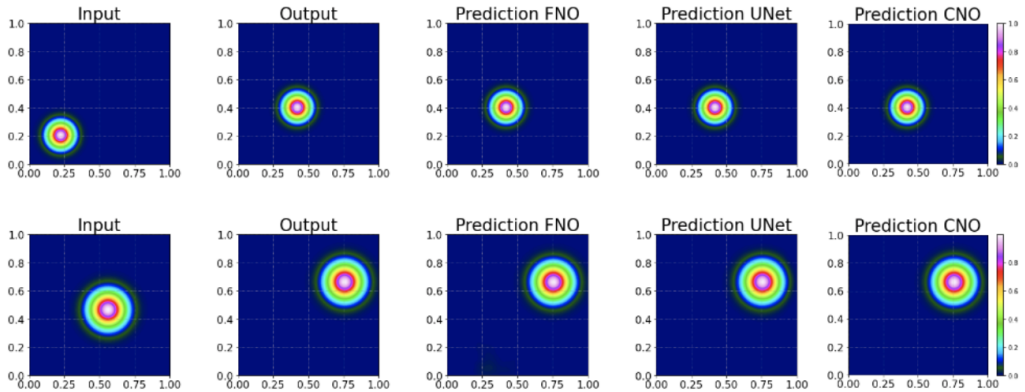


Figure 11: Smooth Transport. Exact and predicted coefficients for an in-distribution (top row) and an out-of-distribution (bottom row) samples and for different models (columns). From left to right: input, output (ground truth), CNO, FNO and UNet.

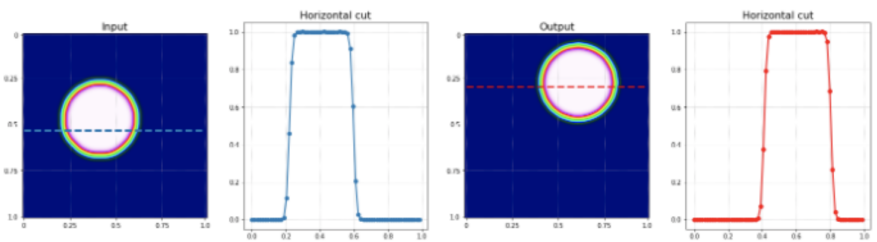


Figure 13: Discontinuous Transport. An example with horizontal cut plots of the disks.

Additional details

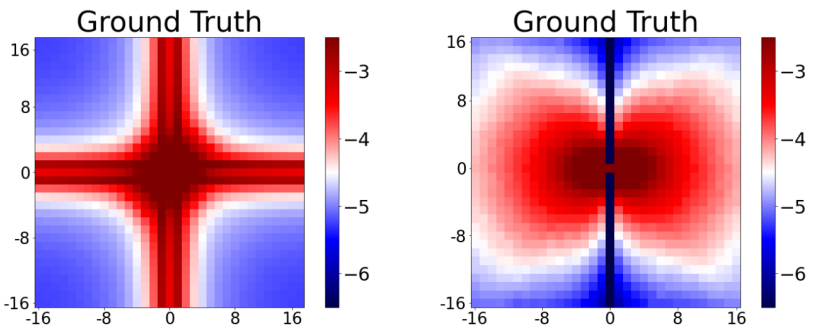


Figure 25: Comparison of the 32 central frequencies of averaged logarithmic amplitude spectra for the two Navier-Stokes experiments. Left: Old NS experiment. Right: Thin shear layer experiment.

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*

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