Deep Learning For Physical Systems

Bogdan Raonić

braonic@ethz.ch

AI4S Symposium, Doha, January 2026

МГ
Математичка гимназија
школа од посебног националног значаја и интереса

l'X
ÉCOLE POLYTECHNIQUE
IP PARIS

ETH zürich

2014-2018 **2018-2021** **2021-2023; 2023 - now**

My group & Collaborators

Prof. Siddhartha Mishra

Prof. Rima Alaifari

Prof. Emmanuel de Bézenac

Prof. Francesca Bartolucci

Dr. Leonardo Zepeda

Dr. Samuel Lanthaler

Dr. Roberto Molinaro

Tobias Rohner

Dr. Tim De Ryck

Maximilian Herde

Levi Lingsch

Yannick Ramic

AI in the Sciences and Engineering (2024)

Description

AI is having a profound impact on science by accelerating discoveries across physics, chemistry, biology, and engineering. This course presents a highly topical selection of AI applications across these fields. Emphasis is placed on using AI, particularly deep learning, to understand systems modelled by PDEs, and key scientific machine learning concepts and themes are discussed.

LINK

README

AI in the Sciences and Engineering, ETH Zurich

LINK

2025

CAMLab ETH zürich

AI in the Sciences and Engineering

HS2025 • ETH Zürich

LINK

Partial Differential Equations

Physics-Informed Neural Networks

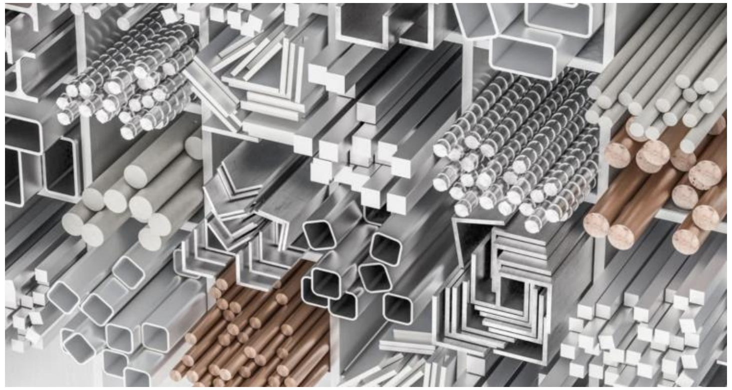
Neural Operators

Foundation Models - Poseidon

Diffusion Models - GenCFD

PDEs

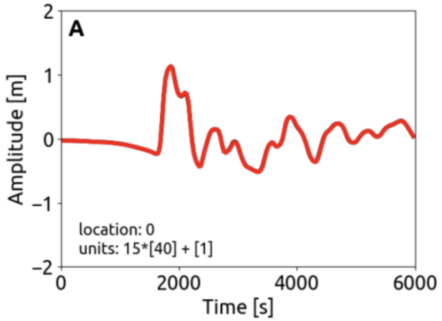
Aircraft modeling

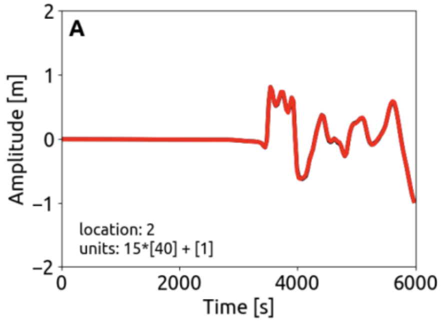


Material modeling

Weather modeling

PDEs



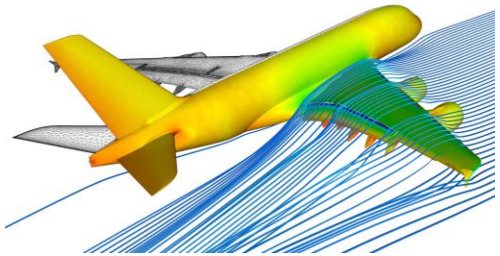


- **Task:** Predict the wave height time series in real time
- Tsunami evacuation

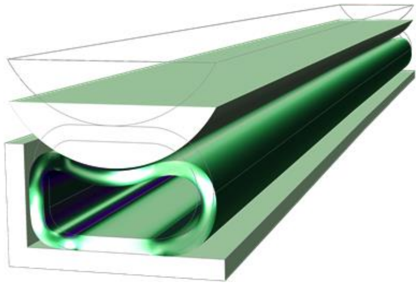
PDEs

STEP 1 – Mathematical Modeling

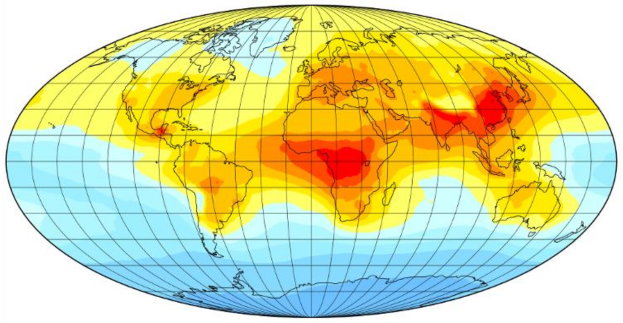
- Model physical phenomena with Partial Differential Equations
- PDEs are **language of nature**



Euler Eqs.



Hyperelastic material Eqs.



Naver-Stokes Eqs. ++

PDEs

Example: **Navier-Stokes equations**

$$\begin{cases} \dfrac{\partial u}{\partial t} + u \cdot \nabla u + \nabla p = \nu \Delta u \\ \nabla \cdot u = 0 \\ u(t=0) = u_0 \end{cases}$$

Weather today

$\mapsto$

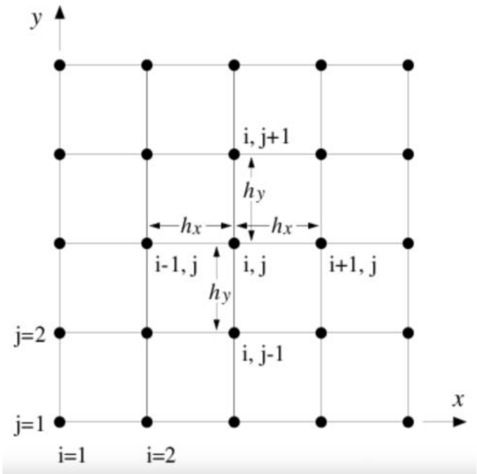
Weather tomorrow

Goal: Approximate **the solution operator** $\mathcal{G}: \omega_0 \mapsto \omega_t$

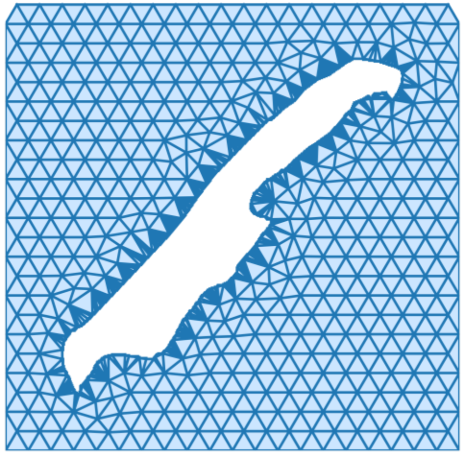
PDEs

STEP 2 – Numerical Simulation

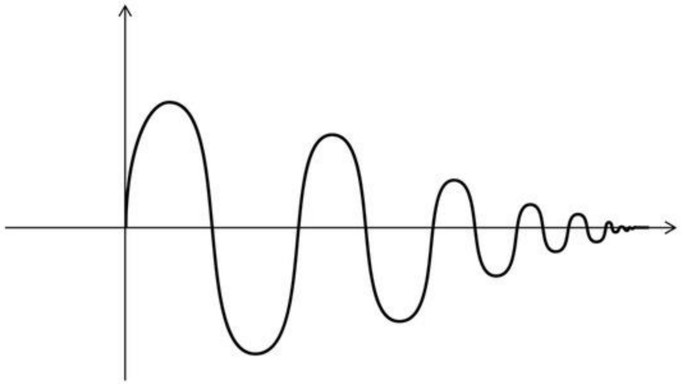
- Analytical (exact) solutions of PDEs are **unknown**
- We use numerical methods to **approximate solutions**



Finite Difference



Finite Element



Spectral

Why ML?

(1) Numerical methods are **manpower intensive**

- Require PhD Level experts

Why ML?

(1) Numerical methods are **manpower intensive**

- Require PhD Level experts

(2) PDE solvers are **compute/time expensive**

- Manu query problems: UQ, Inverse Problems, Design

Why ML?

(1) Numerical methods are **manpower intensive**

- Require PhD Level experts

(2) PDE solvers are **compute/time expensive**

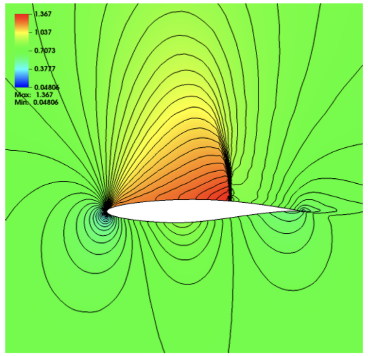
- Manu query problems: UQ, Inverse Problems, Design

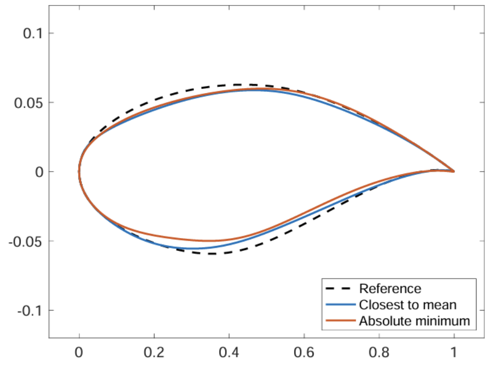
(3) Inapplicable when **physics is missing**

- True for most real-world applications

Why ML?

Example: Constrained Optimization – Airfoil Shape

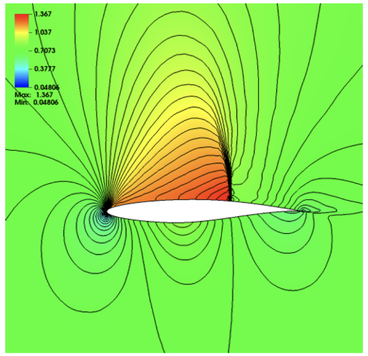


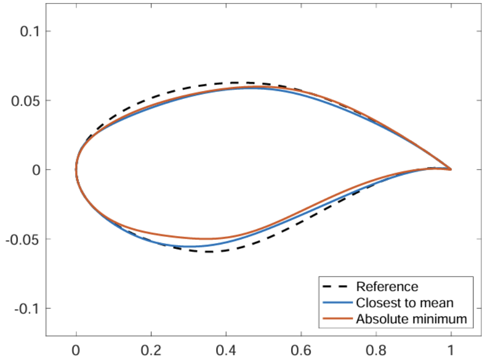


- Parametrized with **Hicks-Henne** shape functions

Why ML?

Example: Constrained Optimization – Airfoil Shape





- Parametrized with **Hicks-Henne** shape functions
- Optimization with Gradient Descent
- At each step: Solve **Euler Equations**
- **Total Computation Time: 228h**

Partial Differential Equations

Physics-Informed Neural Networks

Neural Operators

Foundation Models - Poseidon

Diffusion Models - GenCFD

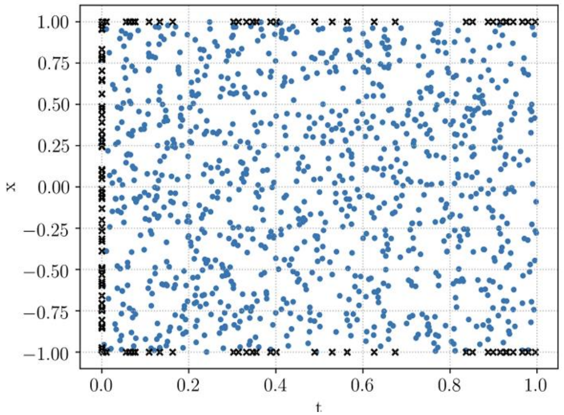
First Attempts - PINNs

Physics-Informed Neural Networks [1]

- Neural Networks that obey **physics constraints**
- **No labeled data**, just physics
- Use **automatic differentiation** for derivatives

First Attempts - PINNs

$$\begin{cases} \frac{\partial u}{\partial t} + u \cdot \nabla u + \nabla p = \nu \Delta u \\ \nabla \cdot u = 0 \\ u(t = 0) = u_0 \end{cases}$$



PDE

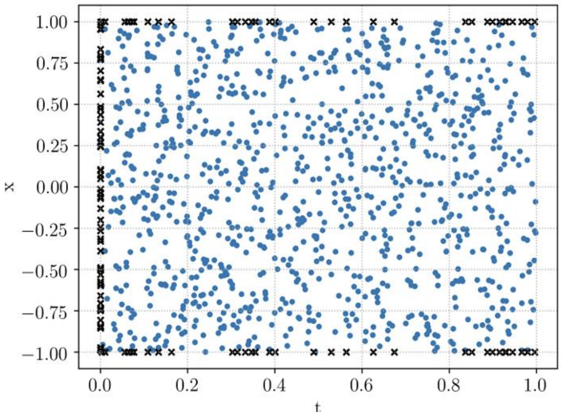
Domain + Points

First Attempts - PINNs

$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p = \nu \Delta \boldsymbol{u} \\ \nabla \cdot \boldsymbol{u} = 0 \\ \boldsymbol{u}(t = 0) = \boldsymbol{u}_0 \end{cases}$$

PDE

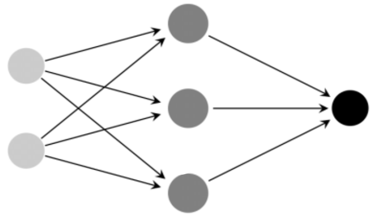
+



Domain + Points

+

$$(x, t) \mapsto u_\theta(x, t), \ \theta \in \Theta$$

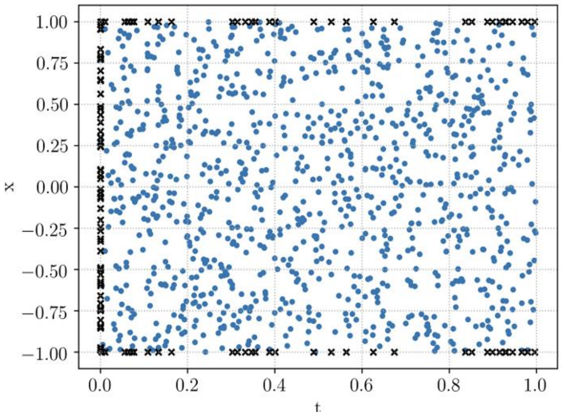


Neural Net

First Attempts - PINNs

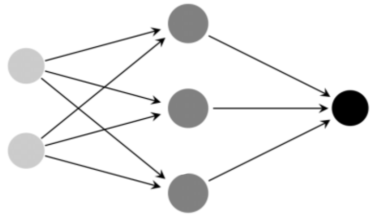
$$\begin{cases} \dfrac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p = \nu \Delta \boldsymbol{u} \\ \nabla \cdot \boldsymbol{u} = 0 \\ \boldsymbol{u}(t=0) = \boldsymbol{u}_0 \end{cases}$$

PDE



Domain + Points

$$(x, t) \mapsto u_\theta(x, t),\ \theta \in \Theta$$



Neural Net

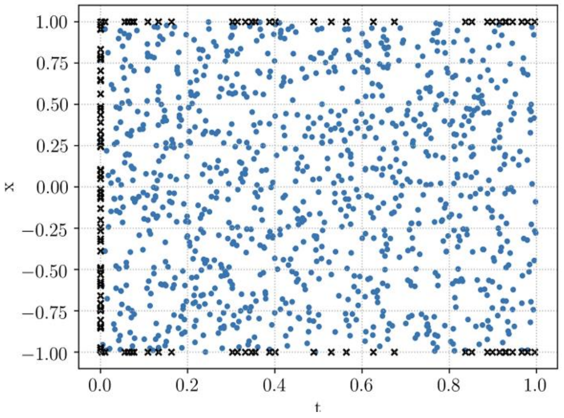
$$J(\theta) = \frac{1}{N_{tb}} \sum_{n=1}^{N_{tb}} |\mathcal{R}_{tb,\theta}(x_n)|^2 + \frac{1}{N_{sb}} \sum_{n=1}^{N_{sb}} |\mathcal{R}_{sb,\theta}(x_n, t_n)|^2 + \frac{1}{N_{int}} \sum_{n=1}^{N_{int}} |\mathcal{R}_{int,\theta}|^2.$$

IC + BC + PDE Loss + AutoDiff

First Attempts - PINNs

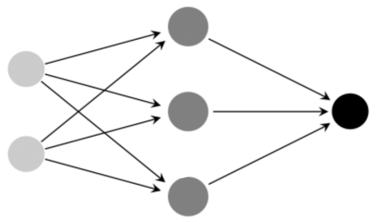
$$
\begin{cases}
\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p = \nu \Delta \boldsymbol{u} \\
\nabla \cdot \boldsymbol{u} = 0 \\
\boldsymbol{u}(t = 0) = \boldsymbol{u}_0
\end{cases}
$$

PDE



Domain + Points

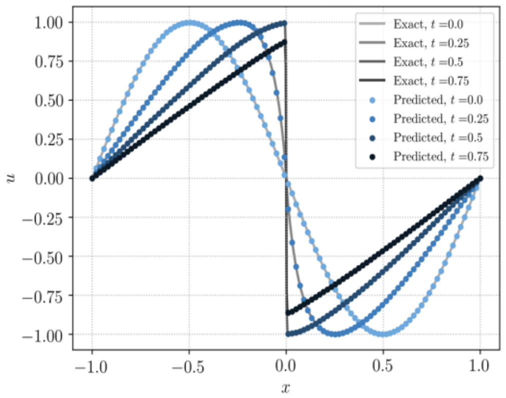
$$(x, t) \mapsto u_\theta(x, t),\ \theta \in \Theta$$



Neural Net

$$J(\theta) = \frac{1}{N_{tb}} \sum_{n=1}^{N_{tb}} |\mathcal{R}_{tb,\theta}(x_n)|^2 + \frac{1}{N_{sb}} \sum_{n=1}^{N_{sb}} |\mathcal{R}_{sb,\theta}(x_n, t_n)|^2 + \frac{1}{N_{int}} \sum_{n=1}^{N_{int}} |\mathcal{R}_{int,\theta}|^2.$$

IC + BC + PDE Loss + AutoDiff

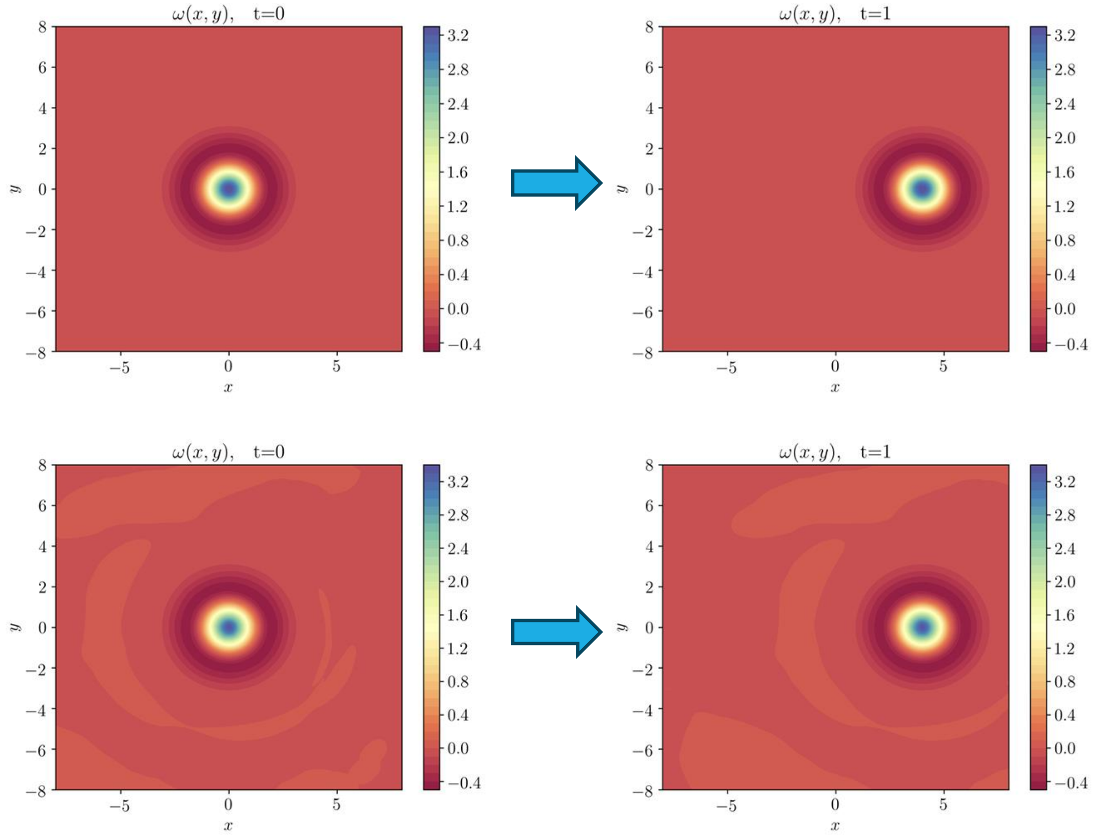


First Attempts - PINNs

Heat Equation

Dimension	Training Error	Total Error
1	2.8×10^{-5}	0.0035%
5	0.0002	0.016%
10	0.0003	0.03%
20	0.006	0.79%
50	0.006	1.5%
100	0.004	2.6%

First Attempts - PINNs

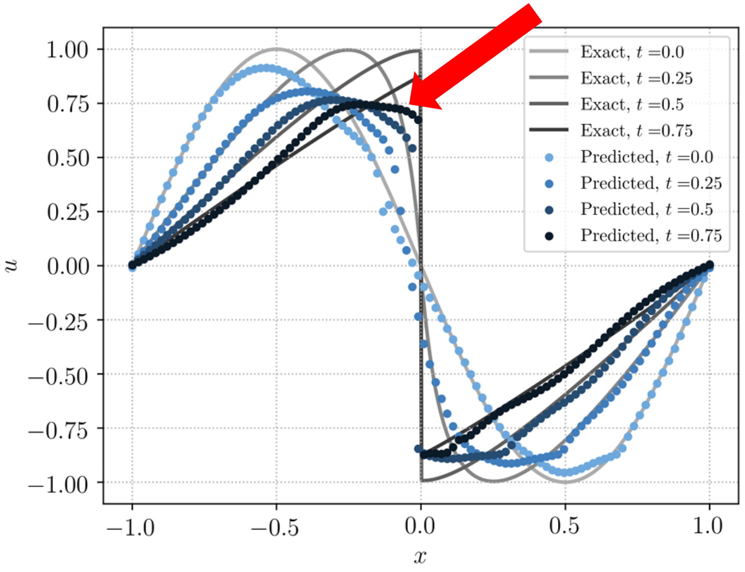


Navier-Stokes Eq.

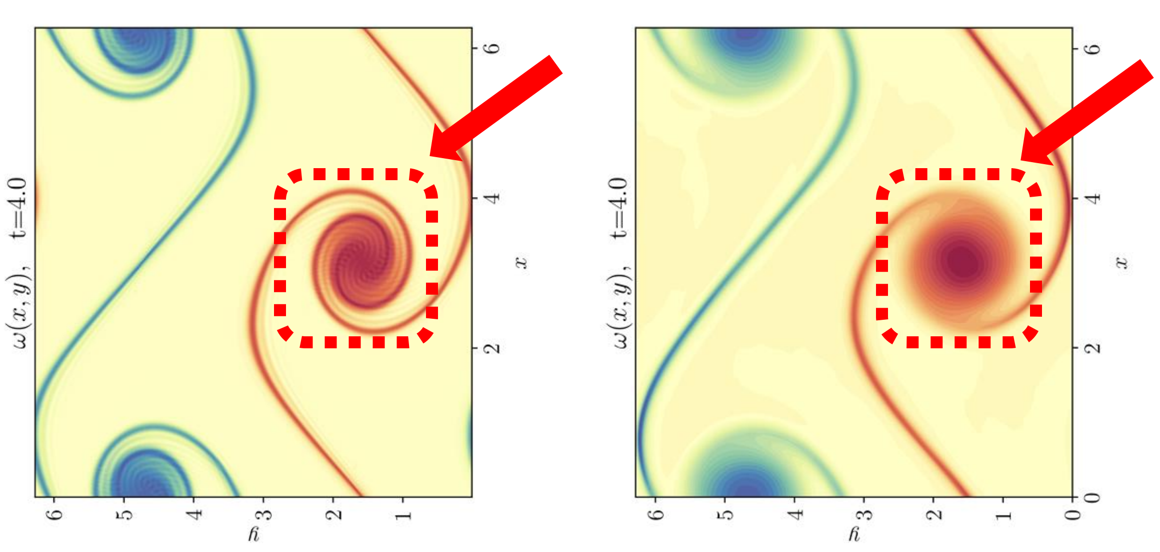
2D Taylor-Vortex

First Attempts - PINNs

- PINNs may not work well when **sharp/strong** gradients are present



Burgers' Equation



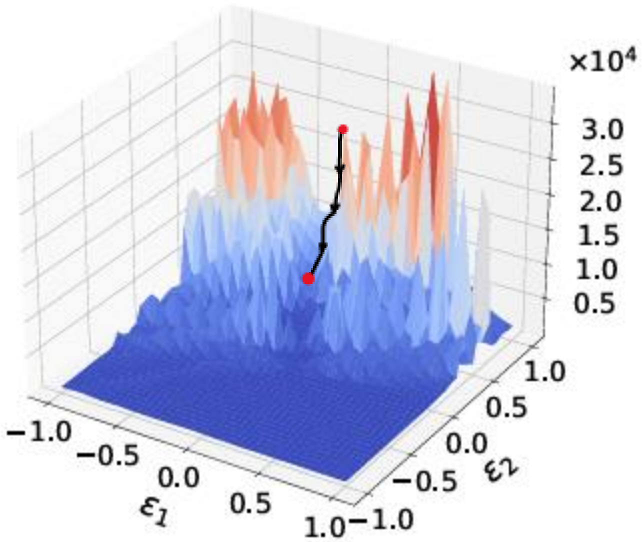
2D Double Shear-Layer

First Attempts - PINNs

- **Poor convergence** – Hard to train

  - Blackbox loss – very high-dimensional non-convex

  - Hard for Nonlinear problems, High-dimensional problems, etc



Loss Landscape
Transport Eq. (high velocity)

First Attempts - PINNs

- Require optimization for each PDE instance

  - Useless for many-query problems

First Attempts - PINNs

- Require optimization for each PDE instance

  - Useless for many-query problems

WE NEED ALTERNATIVES

Partial Differential Equations

Physics-Informed Neural Networks

Neural Operators

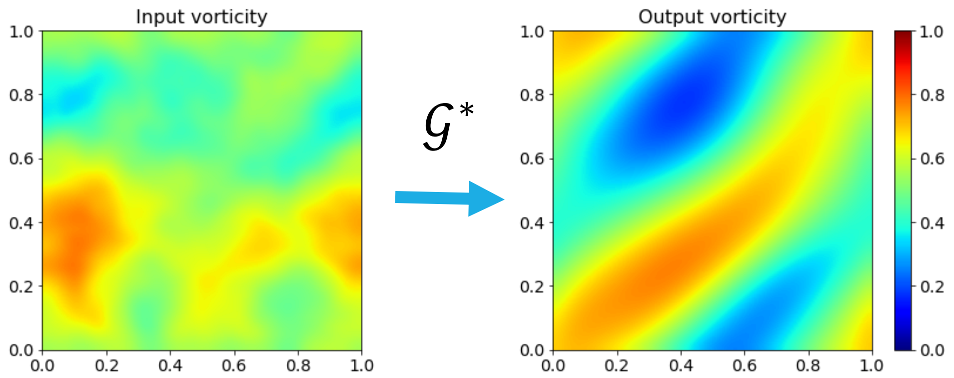
Foundation Models - Poseidon

Diffusion Models - GenCFD

Operator Learning

Supervised Learning

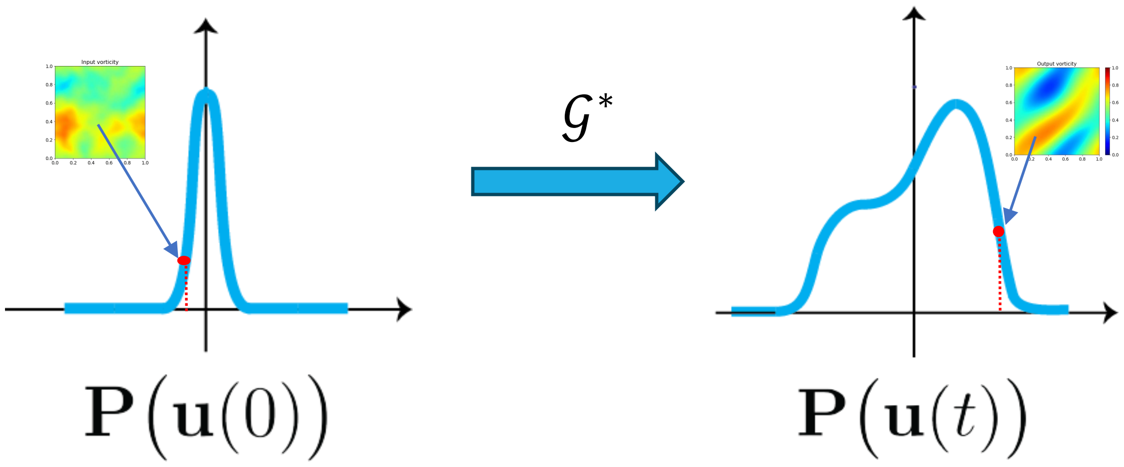
- Construct a **data-driven**, surrogate model for a solution operator
- Data is obtained from **observations** & **simulations**
- Inputs and outputs are functions (in some **discrete** representation)



Neural Operators

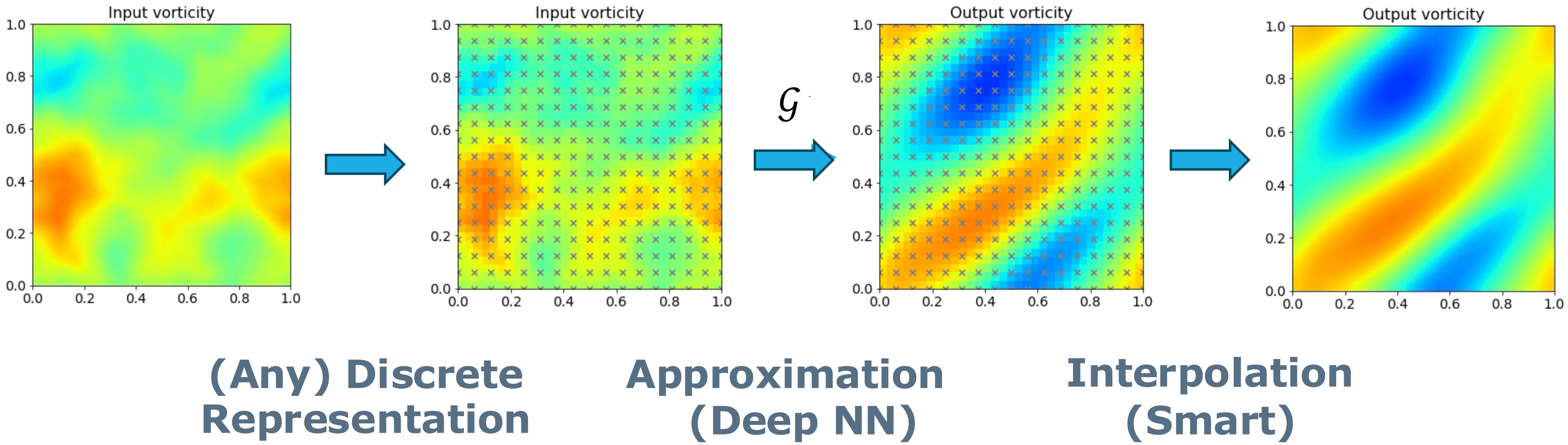
Goal: Find and approximation of G* on data distribution P [2,3]

- Purely data-driven (**black-box**)
- Solve a **family of PDEs** (single set of parameters)
- Possibly slow to train / **Very fast to evaluate**



Neural Operators

How to deal with functions?

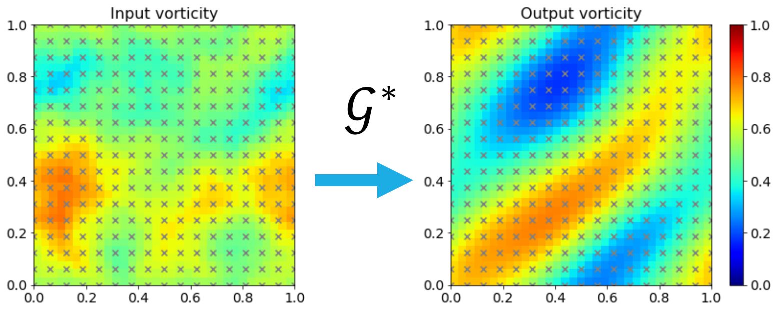


Goal: Learn an underlying **operator** (not just discrete representation)

Pros & Cons

Conventional Numerical Methods

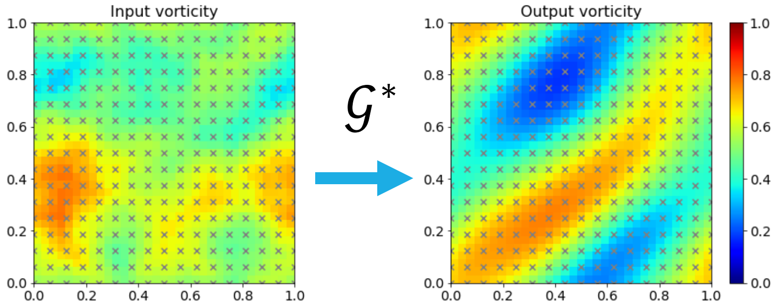
- Require the knowledge of physics (the PDE)
- Solve only one given instance
- Slow on fine grids / Fast on coarse grids
- Solve for **any input parameter**



Pros & Cons

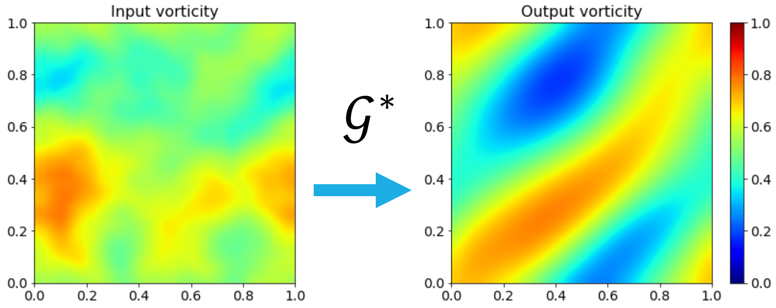
Conventional Numerical Methods

- Require the knowledge of physics (the PDE)
- Solve only one given instance
- Slow on fine grids / Fast on coarse grids
- Solve for **any input parameter**

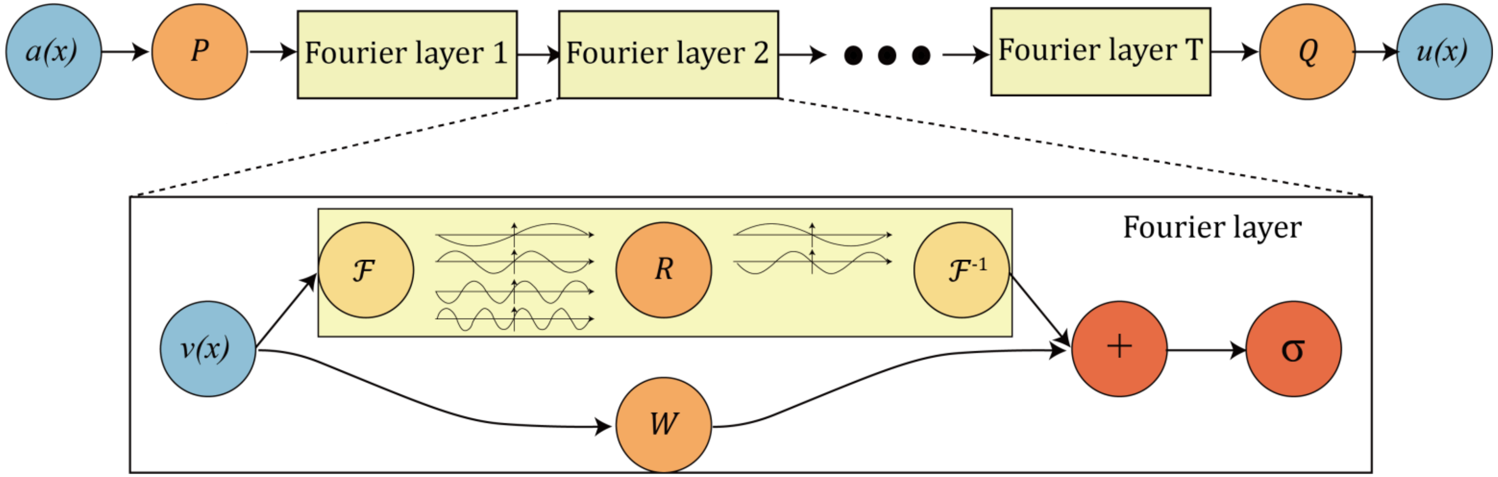


Neural Operators

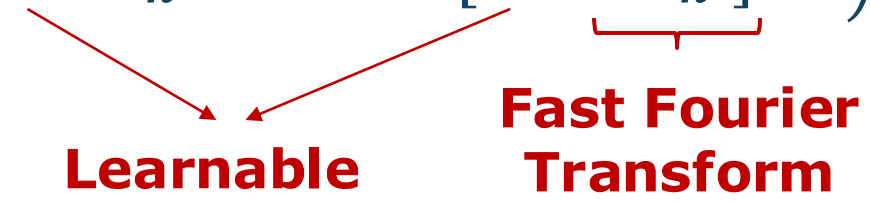
- Purely data-driven (**black-box**)
- Solve a **family of PDEs** (single set of parameters)
- Possibly slow to train / **Very fast to evaluate**
- Solve for parameters from a distribution



Fourier Neural Operators (FNO)

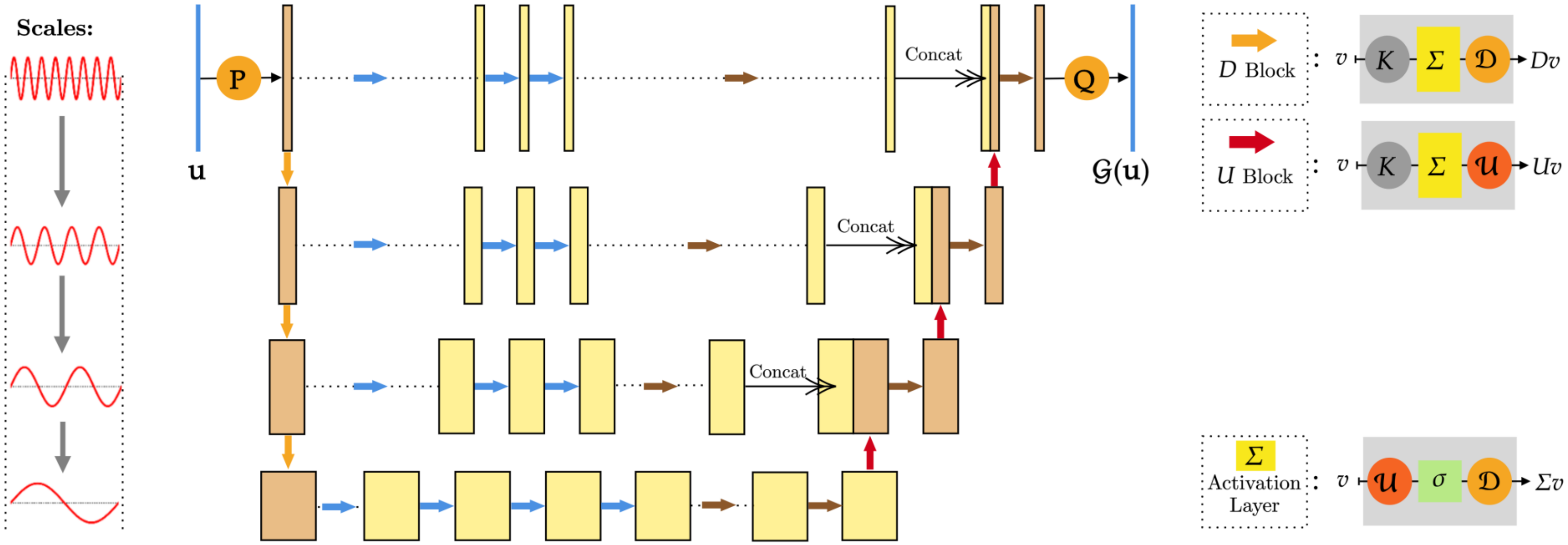


$$v_{n+1}(\mathbf{x}) = \boldsymbol{\sigma}\left(\mathbf{W} \cdot v_n(x) + \mathcal{F}^{-1}\left[R \cdot \mathcal{F}[v_n]\right](x)\right)$$



Convolutional Neural Operators (CNO)

- Multiscale architecture (**fix UNet** for Operator Learning) [4,5]



Convolutional Neural Operators (CNO)

How are operations fixed?

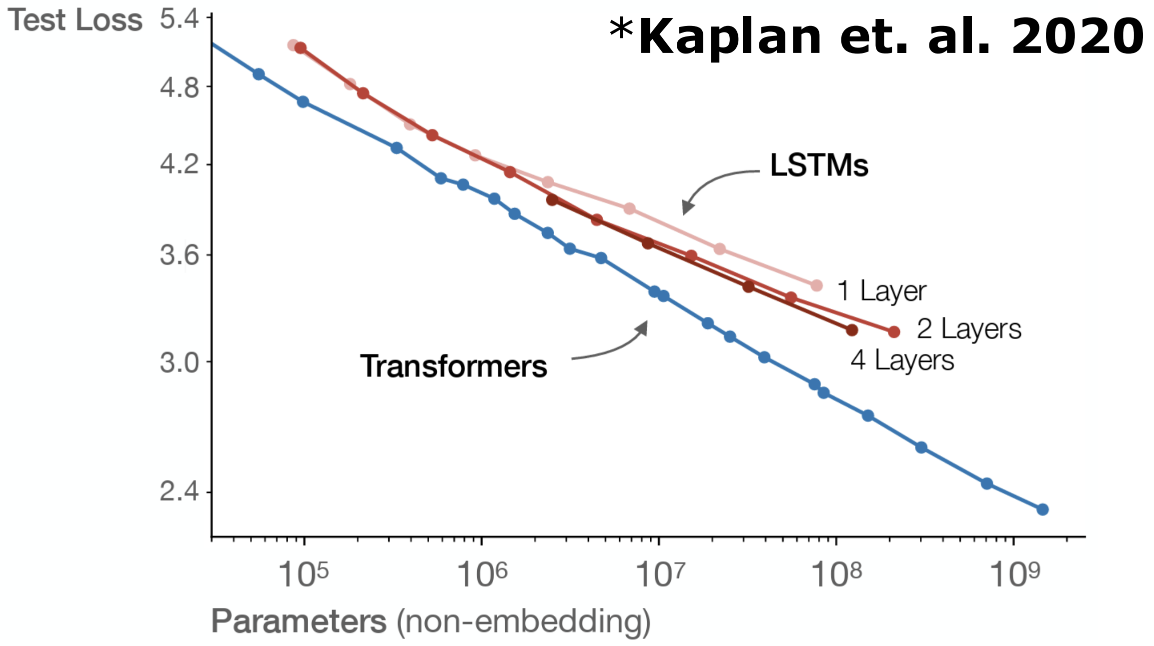
- **Convolution**
- **Up/Downsampling** - Using discrete **sinc interpolation filter**
- **Activation –** Transform it in 3 steps

Transformers

- **Neural Operators** are great for single-task problems with small data size

- They **do not scale** very well with the data

- What happens in language? [8]

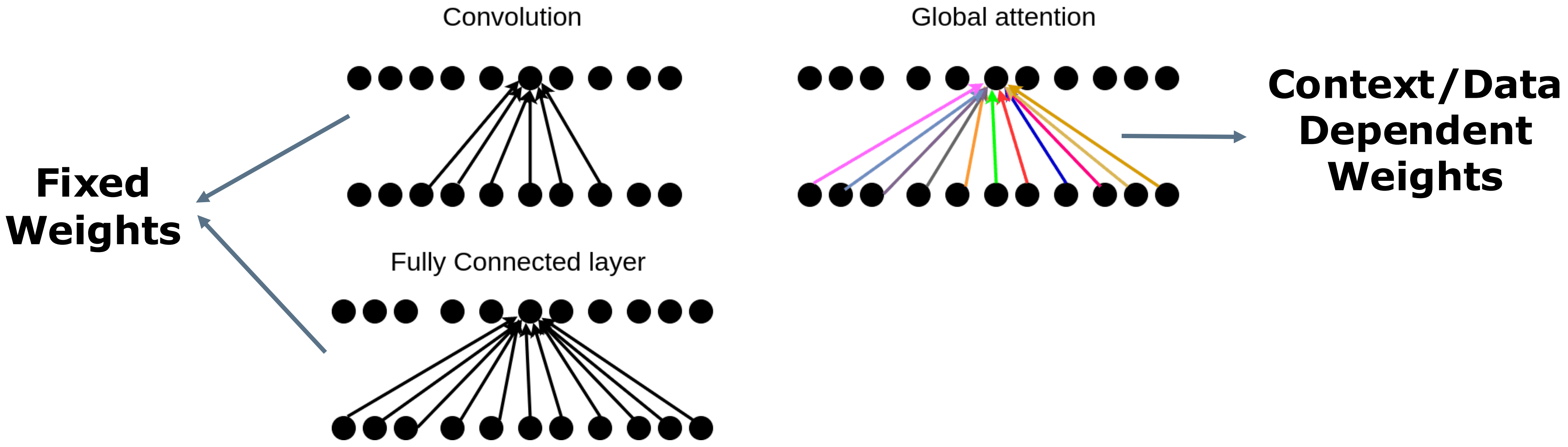


Transformers

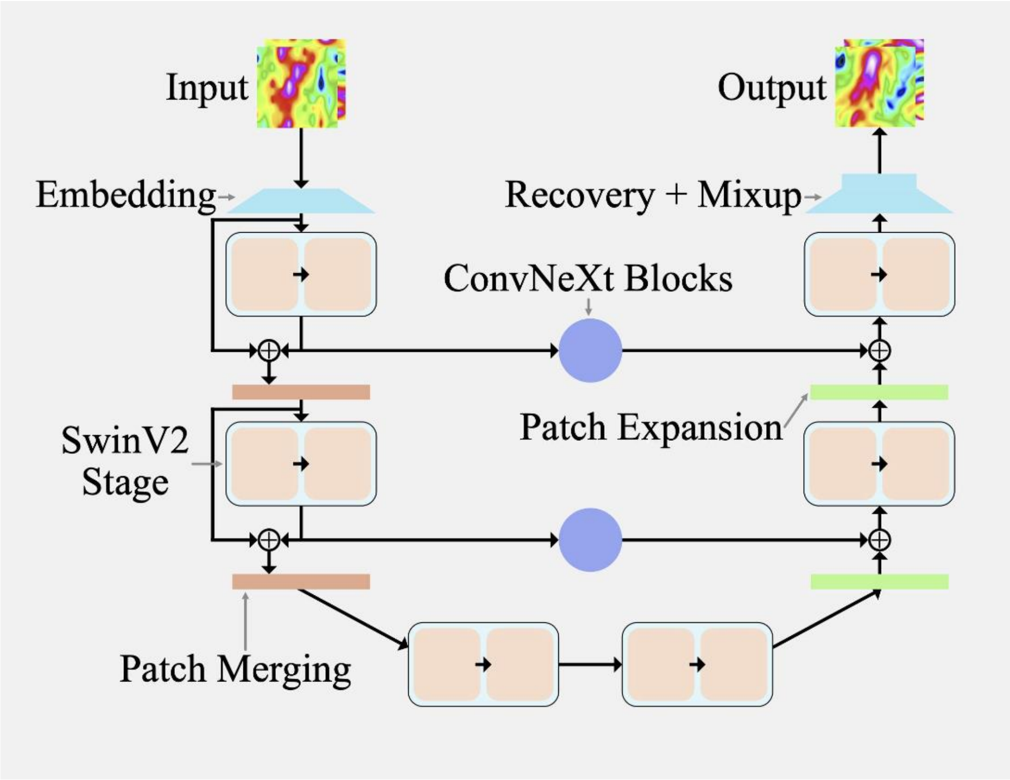
- Transformers process **sequences** [6]
- Core of the Transformers is **attention mechanism**
- Attention ~ The model **focuses** on the most relevant parts of the input
- Decide which **tokens** are important at every layer

Transformers

- Transformers process **sequences** [7]



Scalable Operator Transformer

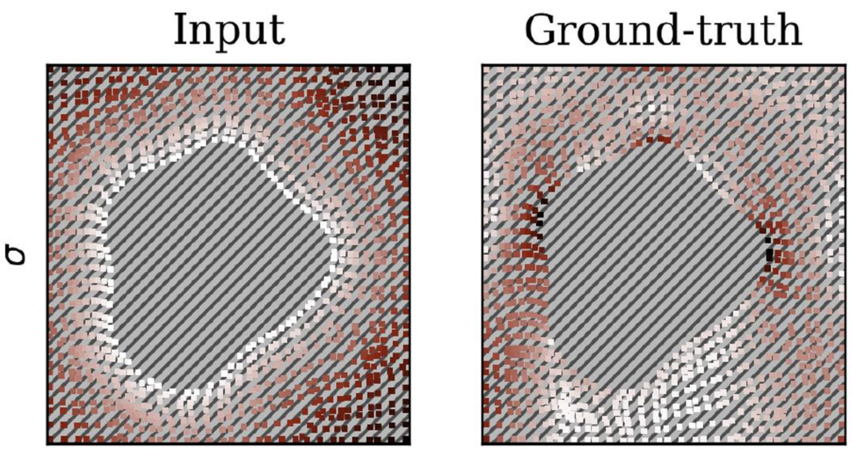


- **Transformers** can be interpreted as Neural Operators
- **Scalable Operator Transformer** (**ScOT**) [10]
- SWIN **Windowed** Attention [12]

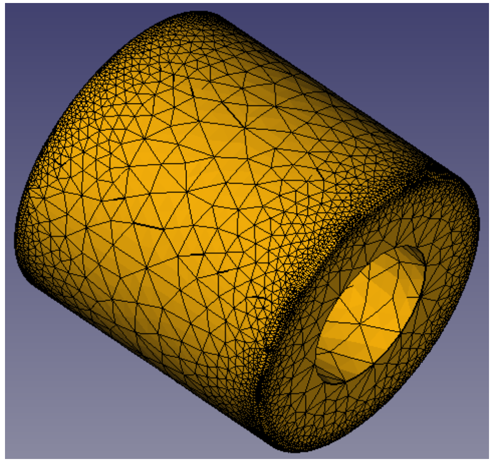
Graph Neural Networks

- Discussion so far has only focused on **Cartesian Domains**
- Discretized with **Uniform Grids**

→ What about **Discretized** data with **Unstructured Grids** or **Point Clouds**

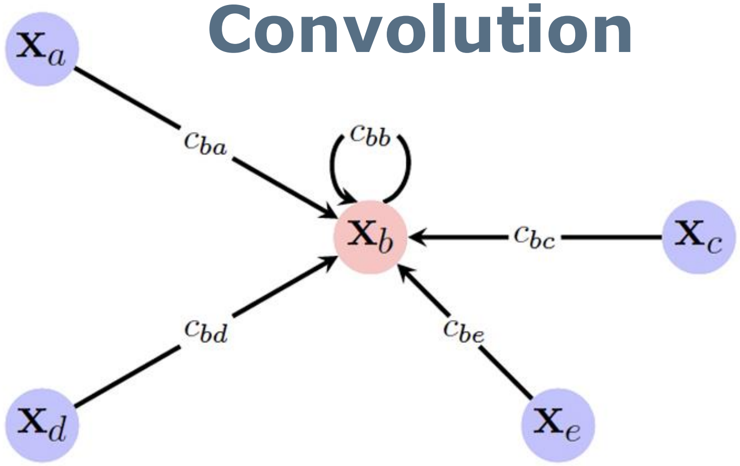


Plasticity



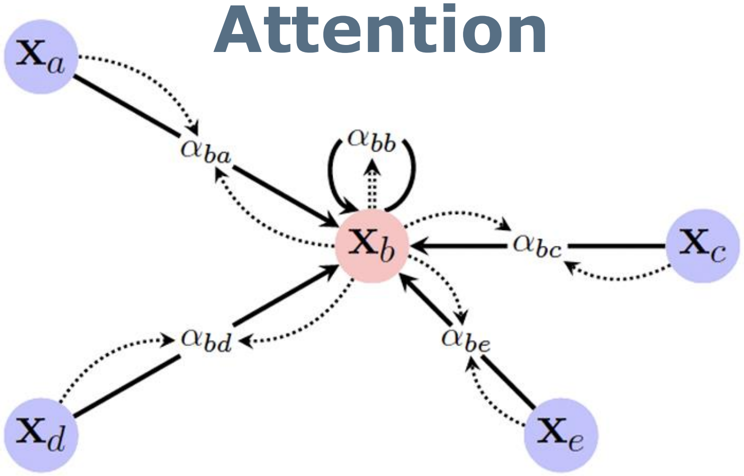
Graph Neural Networks

Graph Convolution



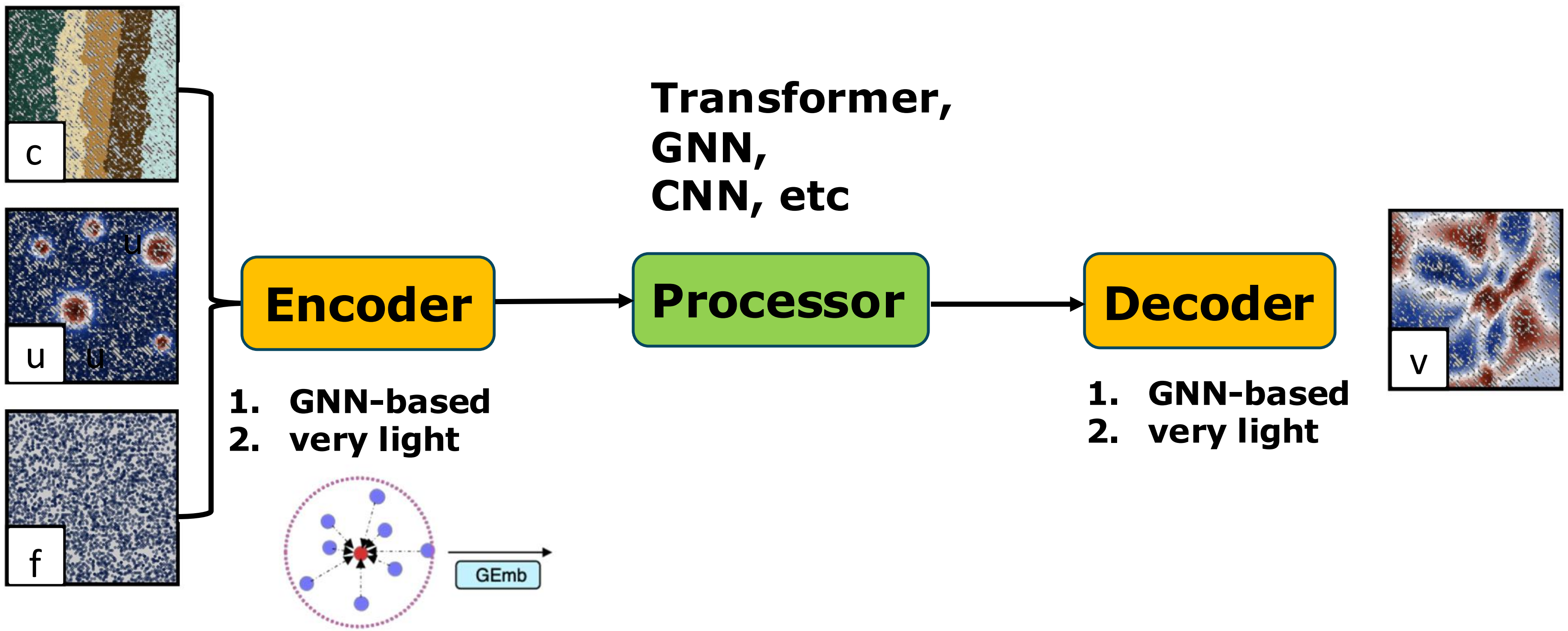
$$\mathsf{h}_i := f\left(\mathsf{v}_i, \bigoplus_{j \in \mathcal{N}_i} c_{ij} \Psi(\mathsf{v}_j)\right)$$

Graph Attention

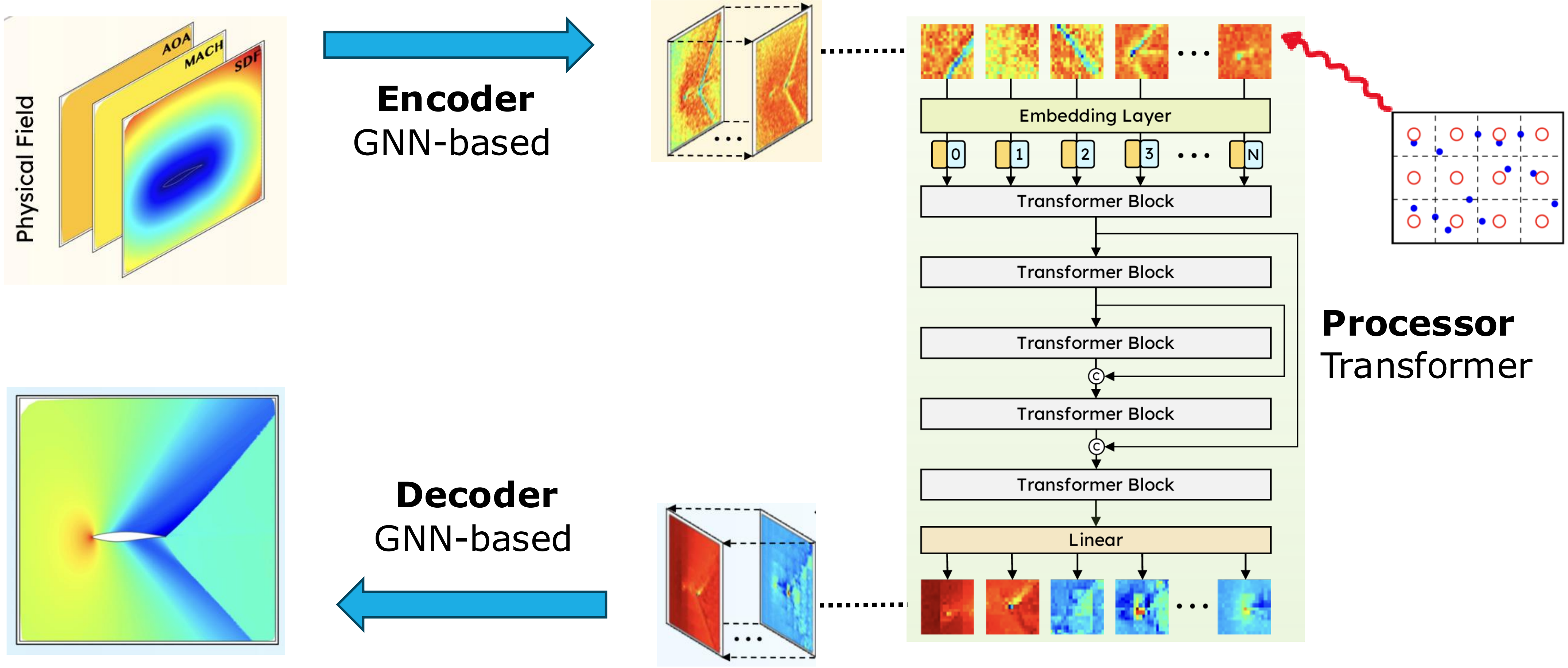


$$\mathsf{h}_i := f\left(\mathsf{v}_i, \bigoplus_{j \in \mathcal{N}_i} \alpha(\mathsf{v}_i, \mathsf{v}_j) \Psi(\mathsf{v}_j)\right)$$

Graph-Based Neural Operator



Geometry Aware Operator Transformer



Partial Differential Equations

Physics-Informed Neural Networks

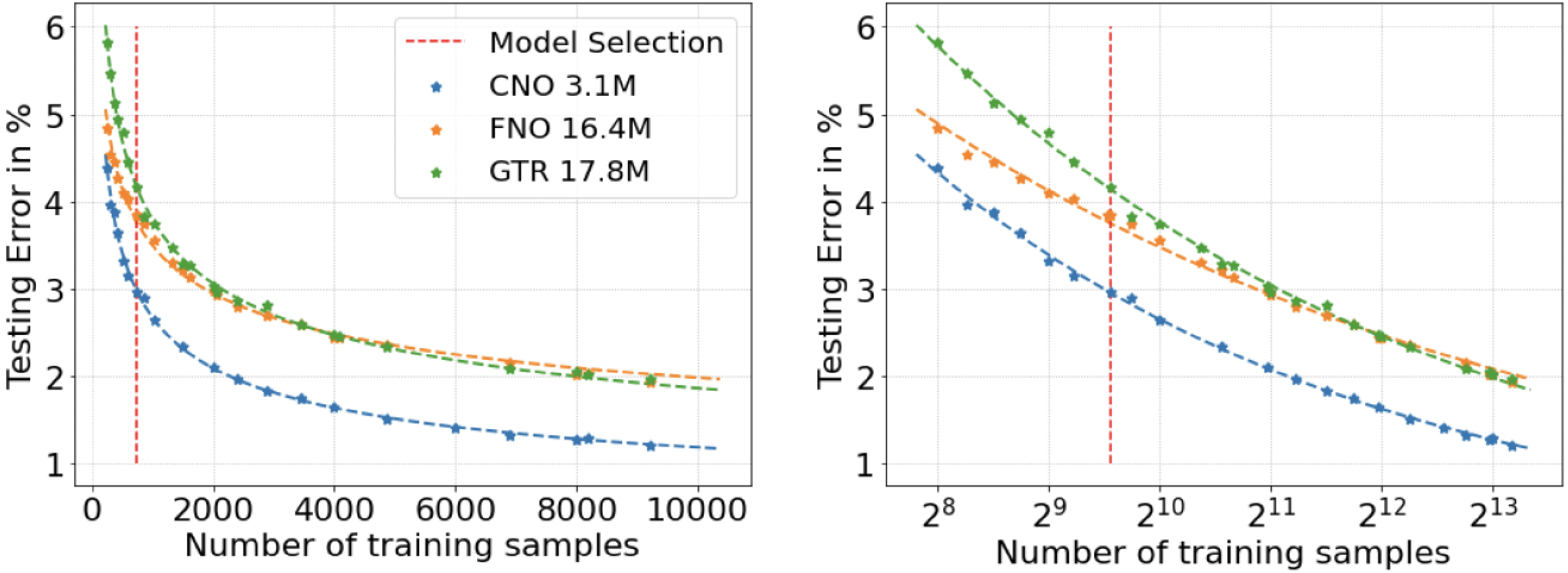
Neural Operators

Foundation Models - Poseidon

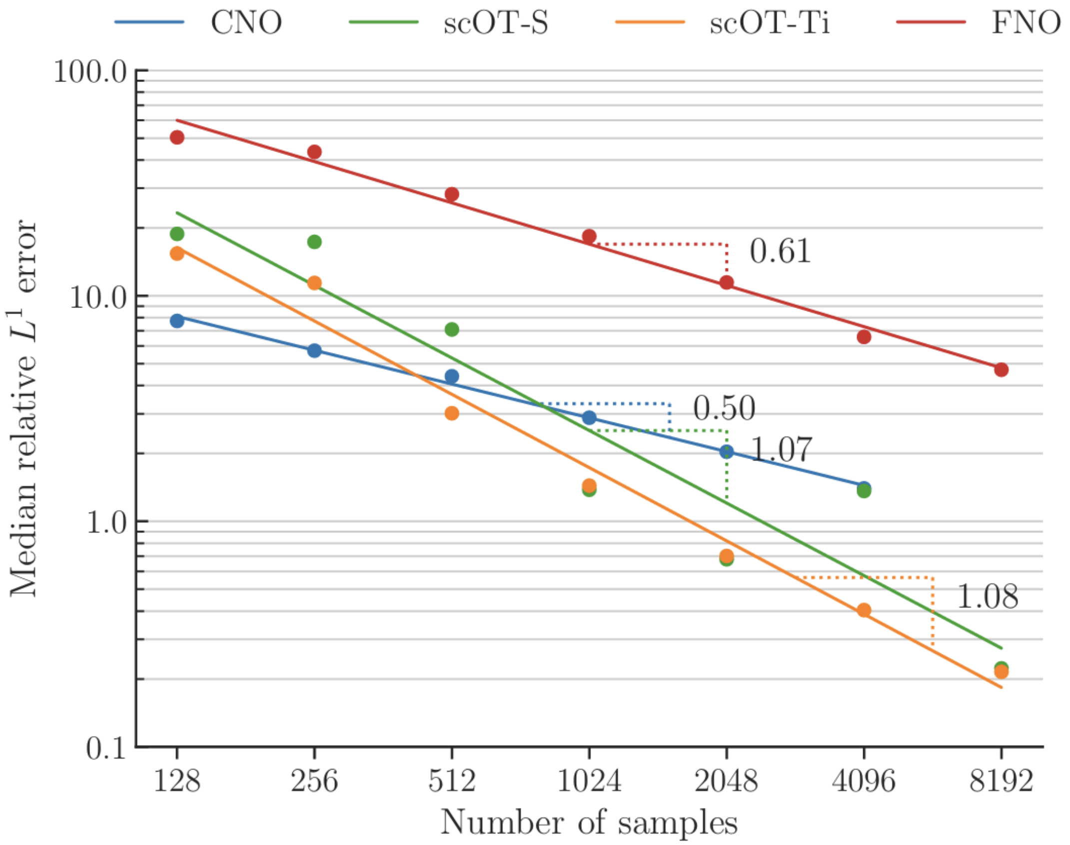
Diffusion Models - GenCFD

Scaling Neural Operators

$\mathcal{E} \sim N^{-\alpha}$ but with α small [4,5,10]



Scalable Operator Transformer



- Still **not great**
- **Large data** is needed

Foundation Models

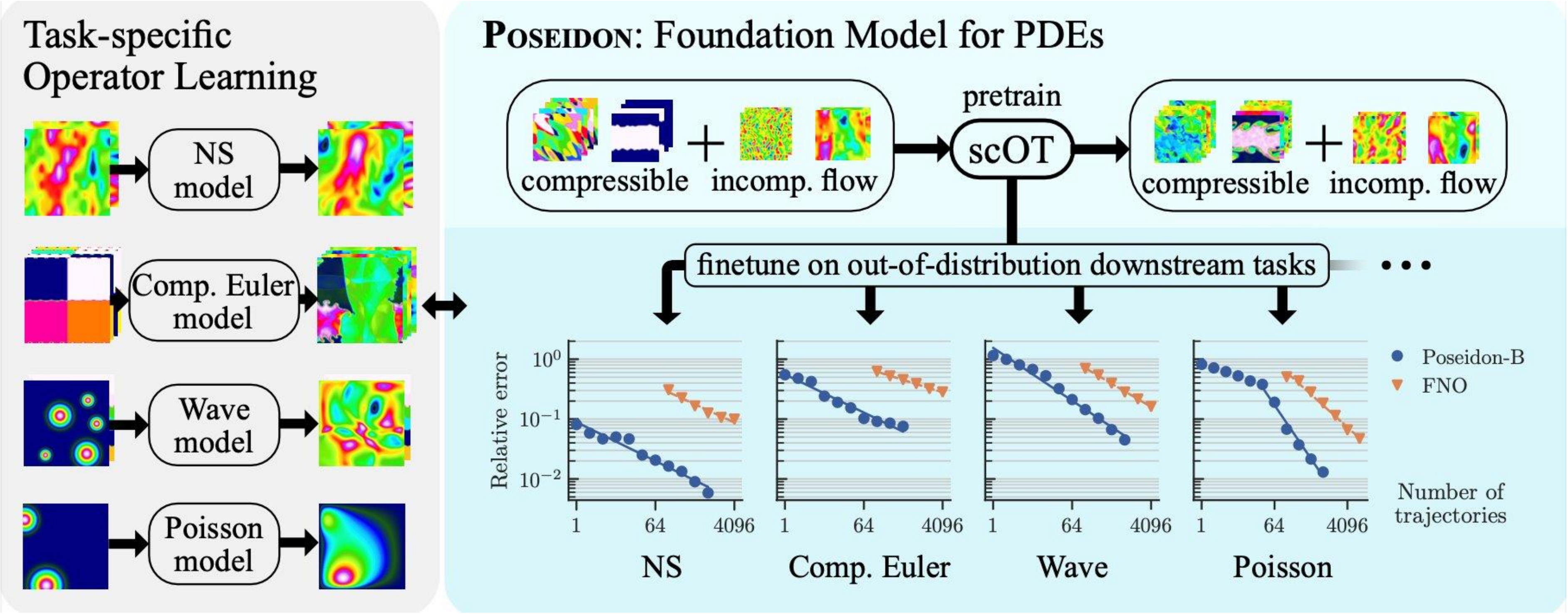
- **Foundation Models** can help?

Step 1: Pretrain a **transformer** model on available data

Step 2: Adapt/Finetune the model on **task specific** data

- Learn effective representations
- Generalize to unseen and unrelated PDEs (via FT)

Poseidon



Task-specific Operator Learning
NS model
Comp. Euler model
Wave model
Poisson model
POSEIDON: Foundation Model for PDEs
compressible
incomp. flow
pretrain
scOT
compressible
incomp. flow
finetune on out-of-distribution downstream tasks
Relative error
10^0
10^{-1}
10^{-2}
1
64
4096
NS
Comp. Euler
Wave
Poisson
Poseidon-B
FNO
Number of trajectories

Poseidon

- Poseidon has an exceptional performance (**SOTA** PDE Foundation Model)

- **Generalizes to new and unseen physics and distributions**

  - Helmholtz

  - Poisson

  - Allen-Cahn …

- **Scales** effectively with model and data size for both

  - Pretraining tasks & Downstream tasks

Partial Differential Equations

Physics-Informed Neural Networks

Neural Operators

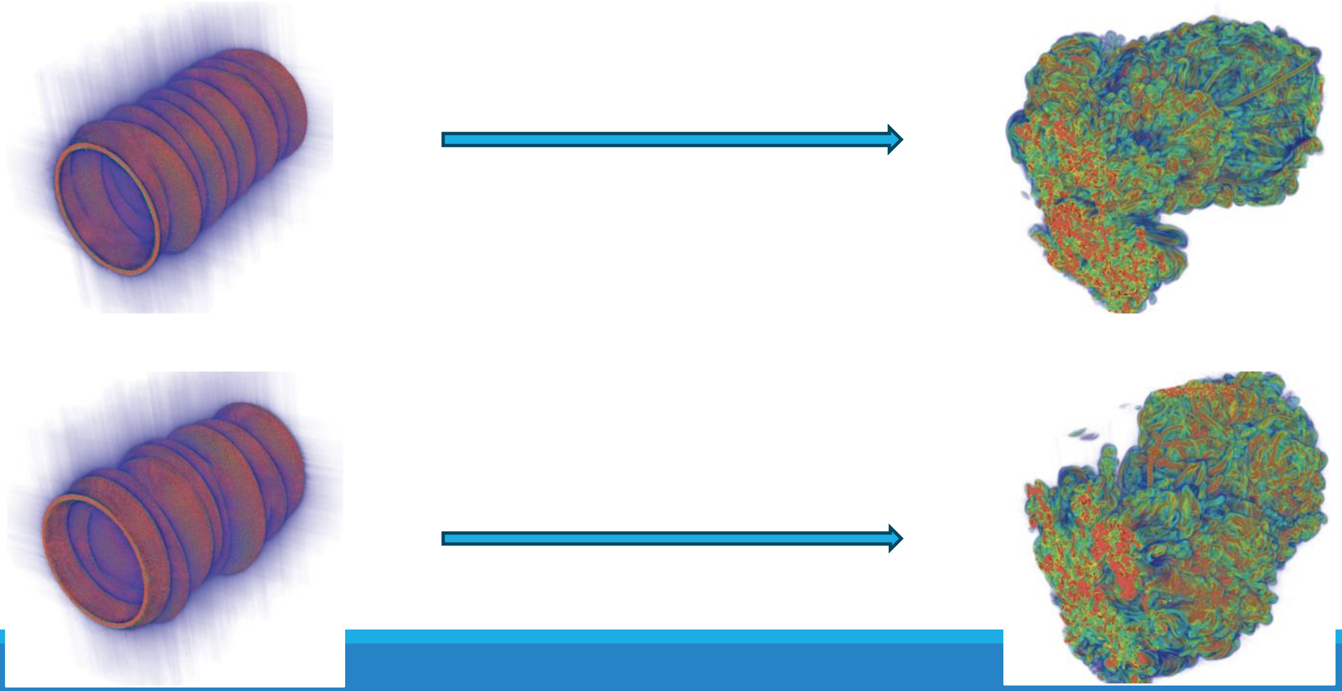
Foundation Models - Poseidon

Diffusion Models - GenCFD

Turbulent Flows

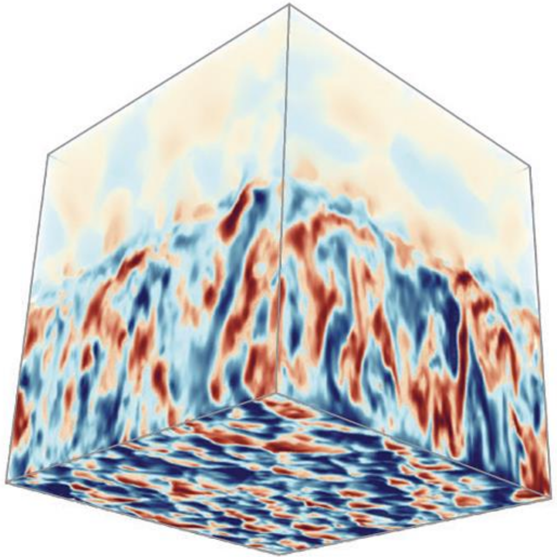
- **Turbulent Flow :** Chaotic nature of the fluid motion

Small changes in the input → **Large** changes in the output

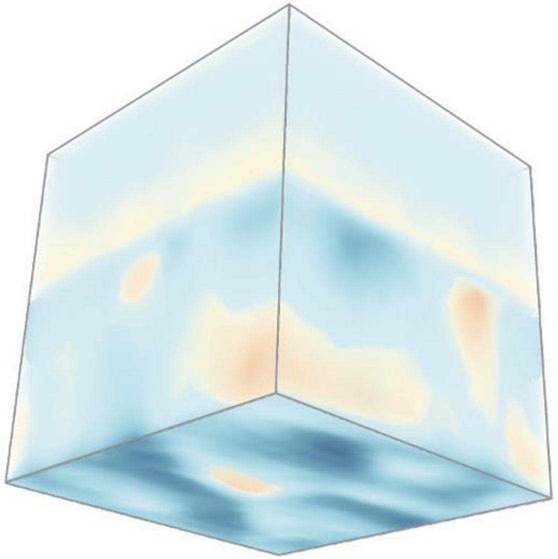


Turbulence Modeling

- Neural Operators cannot approximate **chaotic** solutions of PDEs
- **Lipschitz constants is high**

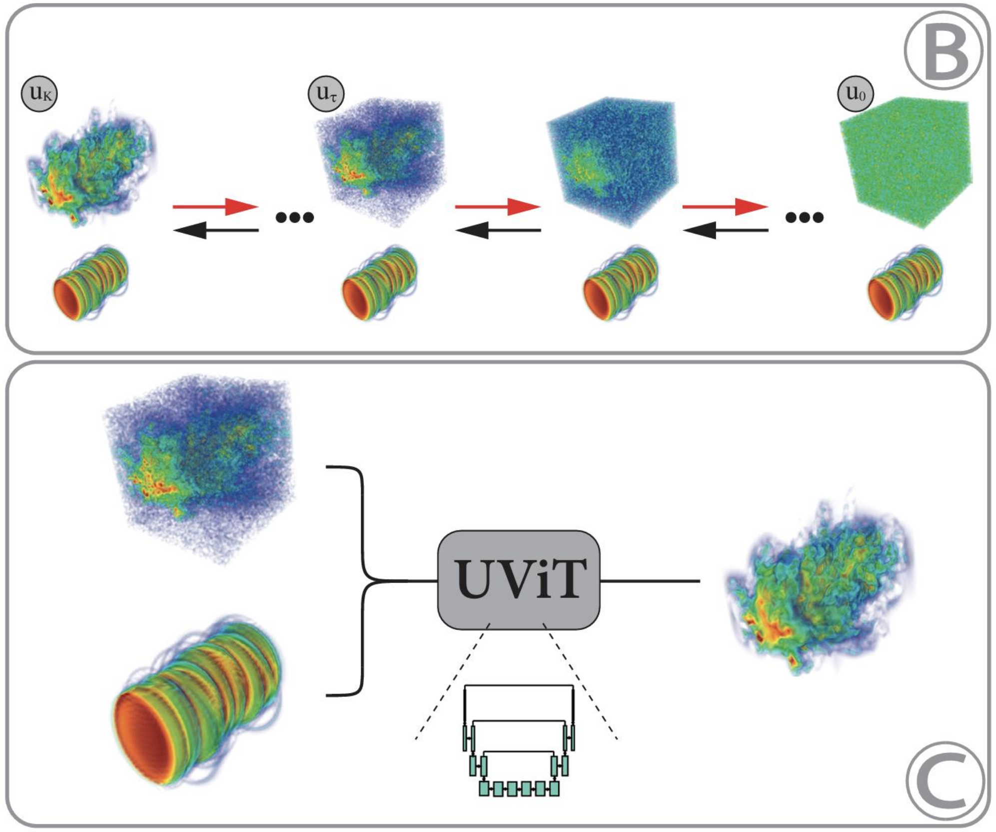


Ground Truth

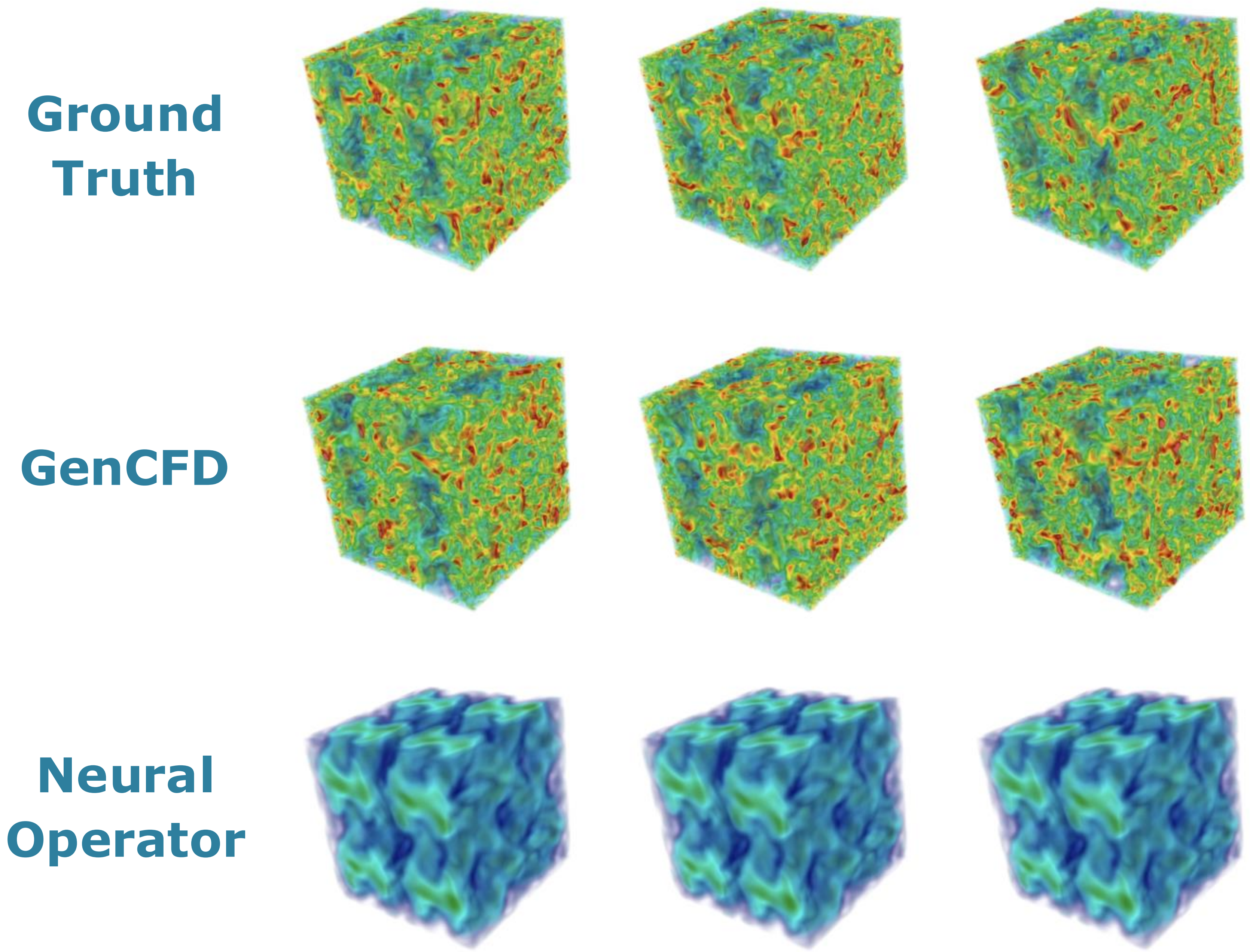


Neural Operator

GenCFD – Diffusion Model



- **Learn** the distribution **from data**
- Conditional Diffusion Models
- **GenCFD** [13]



Samples

Materials and references

[1] Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations - **link**

[2] Neural Operator: Learning Maps Between Function Spaces - **link**

[3] Fourier Neural Operator for Parametric Partial Differential Equations - **link**

[4] Convolutional Neural Operators for robust and accurate learning of PDEs - **link**

[5] Representation Equivalent Neural Operators: a Framework for Alias-free Operator Learning - **link**

[6] Attention Is All You Need - **link**

[7] How Attention works in Deep Learning: understanding the attention mechanism in sequence models - **link**

[8] Scaling Laws for Neural Language Models - **link**

[9] An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale - **link**

[10] 🧜‍♂️ Poseidon: Efficient Foundation Models for PDEs - **link**

[11] Vision Transformers (ViT) Explained - **link**

[12] Swin Transformer: Hierarchical Vision Transformer using Shifted Windows - **link**

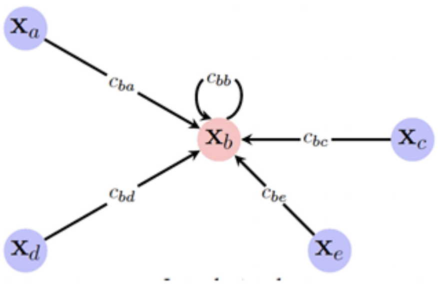
[13] Generative AI for fast and accurate statistical computation of fluids - **link**

Thank you for your attention

Towards COLAB

braonic.ethz.ch

bogdan.raonke@gmail.com



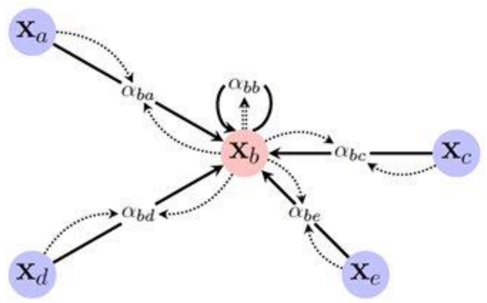
- Generic form of GCN:

$$\mathrm{h}_i := f\left(\mathrm{v}_i, \bigoplus_{j\in\mathcal{N}_i} c_{ij}\Psi(\mathrm{v}_j)\right)$$

- Specific example:

$$\mathrm{h}_i := g\left(\sum_{j\in\mathcal{N}_i\cup\{v_i\}} \frac{1}{\sqrt{\tilde{d}_i\tilde{d}_j}}\mathrm{W}\mathrm{v}_j\right), \quad \tilde{d}_i = 1 + \sum_j \mathrm{A}_{ij}$$

$$\tilde{L}_{\text{sym}} = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}, \quad \tilde{A} = A + I, \quad \tilde{D}_{ii} = \Sigma_j \tilde{A}_{ij}$$



- Generic form of GAT:

$$\mathrm{h}_i := f\left(\mathrm{v}_i, \bigoplus_{j \in \mathcal{N}_i} \alpha(\mathrm{v}_i, \mathrm{v}_j)\Psi(\mathrm{v}_j)\right)$$

- Specific example:

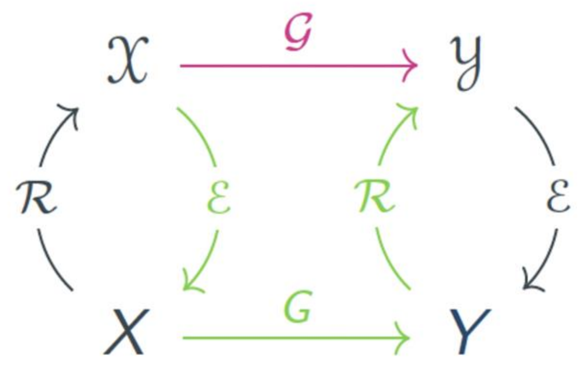
$$e_{ij} = a\left(W\boldsymbol{h}_i, W\boldsymbol{h}_j\right) \qquad \alpha_{ij} = \operatorname{soft}\max_j\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{v_k \in \tilde{N}(v_i)} \exp\left(e_{ik}\right)}$$

$$\mathrm{h}_i := g\left(\sum_{j \in \mathcal{N}_i \cup \{\mathrm{v}_i\}} \alpha(\mathrm{v}_i, \mathrm{v}_j)\mathrm{W}\mathrm{v}_j\right),$$

Neural Operators

Continuous Spaces

Discrete Spaces



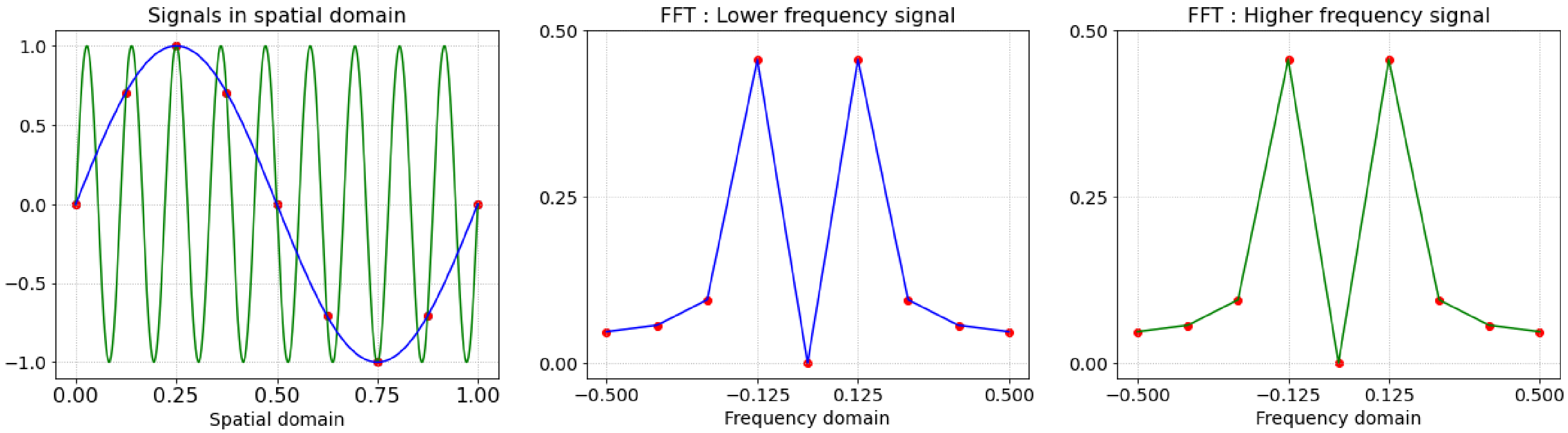
$$\mathcal{G} = \mathcal{R} \circ G \circ \mathcal{E}$$

Design your model in such a way that [4,5]

Learning at the discrete level = Learning at the continuous level

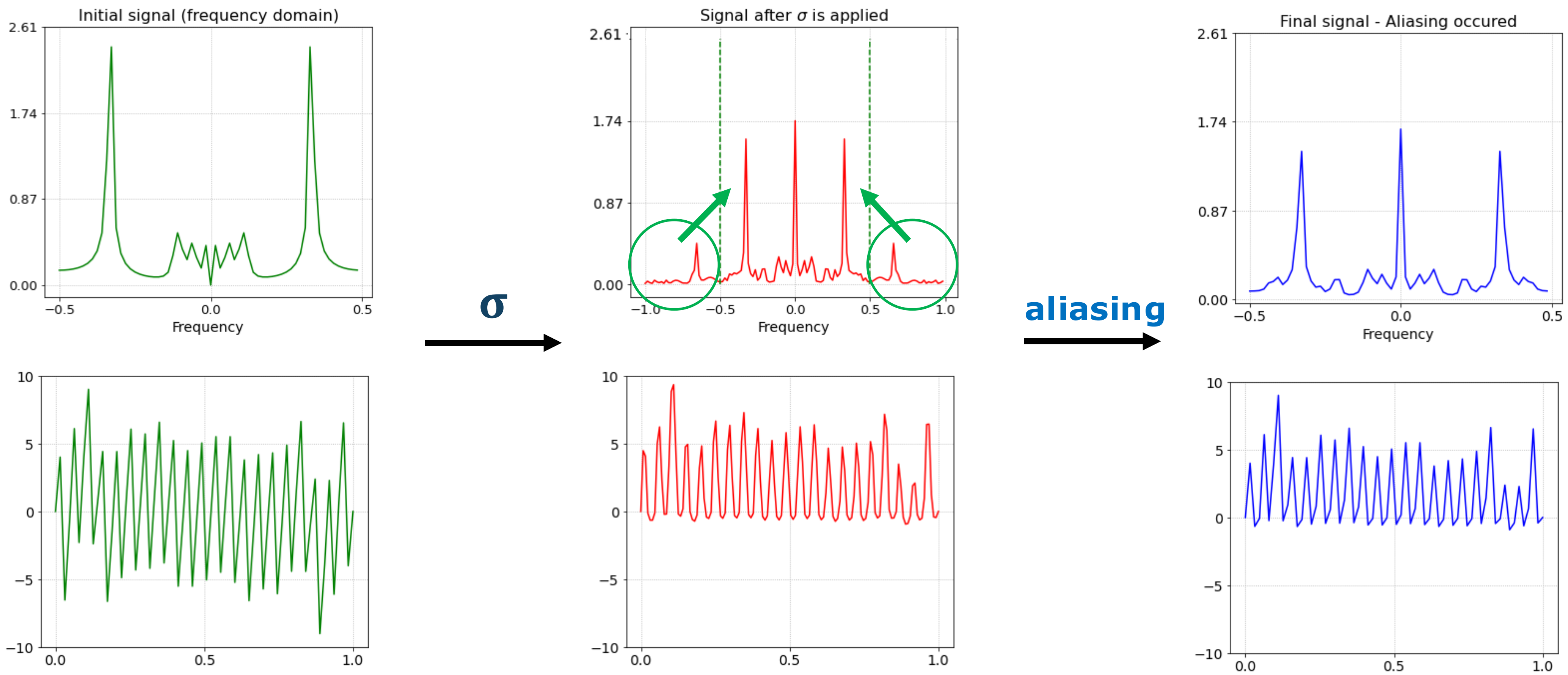
Neural Operators

Why **ALL** operations?



Aliasing Effect – **CDE** is broken

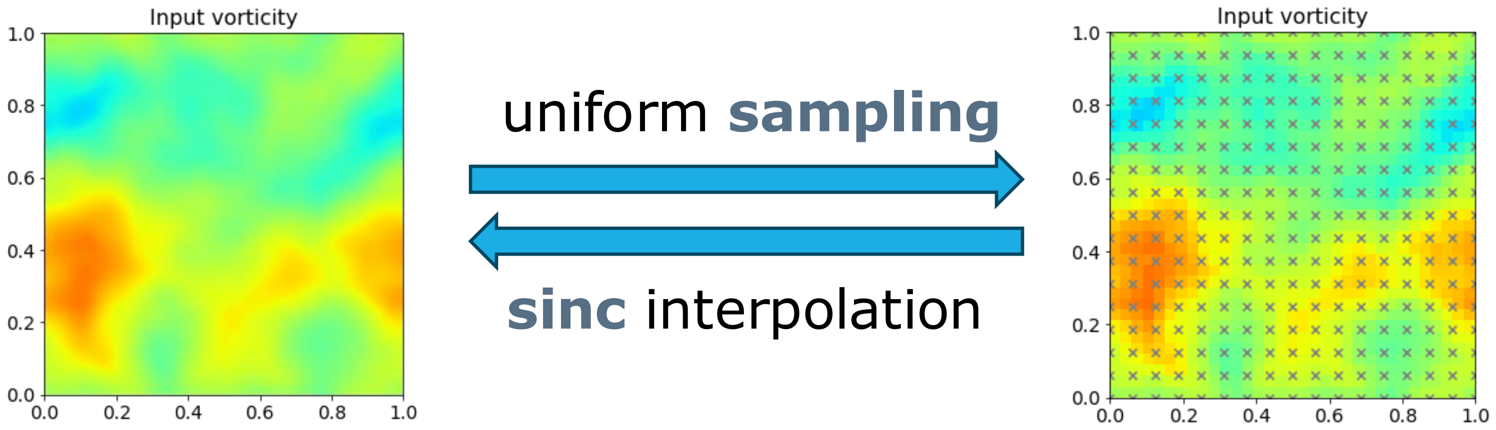
Neural Operators



Neural Operators

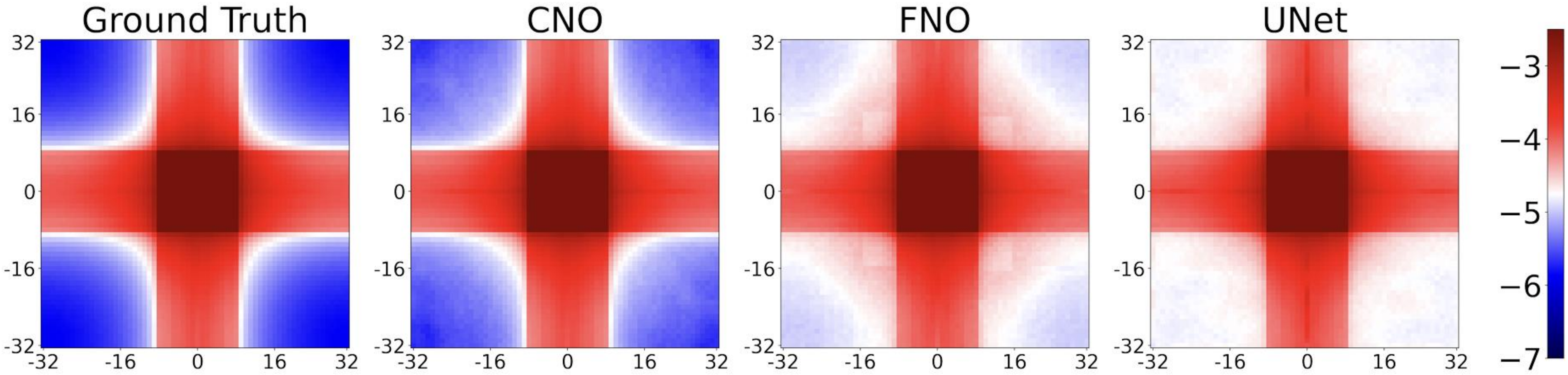
Bandlimited function

- **Bounded** frequency spectrum
- Natural CDE property



Results

2D Poisson Equation



Results

- Testing **CDE** property