Universität Bremen

Center for Industrial Mathematics (ZeTeM)

Faculty 03

Mathematics / Computer science

Operator Learning and DeepONets

Janek Gödeke, Nick Heilenkötter, Tom Freudenberg

Heidelberg, 08.11.2023

Yesterday: PINNs for Single PDE

Poisson equation:

$$\begin{aligned} \Delta u(x) &= f(x) \quad && \text{on } \Omega \subset \mathbb{R}^2, \\ u(x) &= u_0 && \text{for } x \in \partial\Omega. \end{aligned}$$

Task: Learn u for **fixed** $u_0 \in \mathbb{R}$ and $f : \Omega \to \mathbb{R}$

Yesterday: PINNs for Single PDE

Poisson equation:

$$\begin{aligned} \Delta u(x) &= f(x) \quad &&\text{on } \Omega \subset \mathbb{R}^2, \\ u(x) &= u_0 \quad &&\text{for } x \in \partial\Omega. \end{aligned}$$

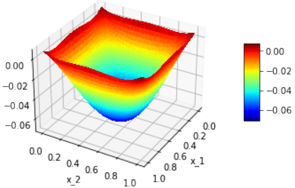


Figure: Learned solution for $f = 1$ and $u_0 = 0$

Task: Learn u for **fixed** $u_0 \in \mathbb{R}$ and $f : \Omega \to \mathbb{R}$

Yesterday: PINNs for Single PDE

Poisson equation:

$$\begin{aligned} \Delta u(x) &= f(x) \quad && \text{on } \Omega \subset \mathbb{R}^2, \\ u(x) &= u_0 && \text{for } x \in \partial\Omega. \end{aligned}$$

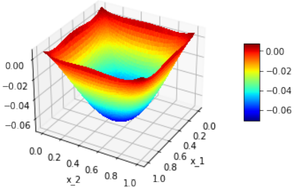


Figure: Learned solution for $f = 1$ and $u_0 = 0$

Task: Learn u for **fixed** $u_0 \in \mathbb{R}$ and $f : \Omega \to \mathbb{R}$

Neural network u_θ

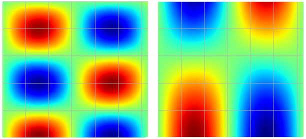
$$u_\theta : \Omega \longrightarrow \mathbb{R}$$

$$u_\theta(x) \approx u(x)$$

This Morning: PINNs for Multiple PDEs

Wave equation:

$$
\begin{aligned}
\partial_t^2 u &= c^2 \partial_x^2 u && \text{on } I_x \times I_t \subset \mathbb{R}^2, \\
u &= 0 && \text{on } \partial I_x \times I_t \\
\partial_t u(\cdot, 0) &= 0 && \text{on } I_x \\
u(x, 0) &= f(x) && \text{for } x \in I_x.
\end{aligned}
$$

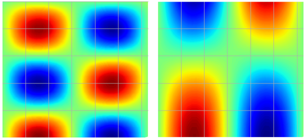


Task: Learn u_c for **all** $c \in I_c$
fixed $f : I_x \to \mathbb{R}$

This Morning: PINNs for Multiple PDEs

Wave equation:

$$
\begin{aligned}
\partial_t^2 u &= c^2 \partial_x^2 u && \text{on } I_x \times I_t \subset \mathbb{R}^2,\\
u &= 0 && \text{on } \partial I_x \times I_t\\
\partial_t u(\cdot, 0) &= 0 && \text{on } I_x\\
u(x, 0) &= f(x) && \text{for } x \in I_x.
\end{aligned}
$$



Task: Learn u_c for **all** $c \in I_c$

fixed $f : I_x \to \mathbb{R}$

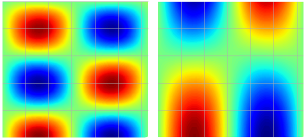
Function to be learned

$$
\begin{aligned}
I_x \times I_t \times I_c &\longrightarrow \mathbb{R}\\
(x, t, c) &\longmapsto u_c(x, t)
\end{aligned}
$$

This Morning: PINNs for Multiple PDEs

Wave equation:

$$
\begin{aligned}
\partial_t^2 u &= c^2 \partial_x^2 u && \text{on } I_x \times I_t \subset \mathbb{R}^2,\\
u &= 0 && \text{on } \partial I_x \times I_t\\
\partial_t u(\cdot, 0) &= 0 && \text{on } I_x\\
u(x,0) &= f(x) && \text{for } x \in I_x.
\end{aligned}
$$



Task: Learn u_c for **all** $c \in I_c$

fixed $f : I_x \to \mathbb{R}$

Function to be learned

$$
\begin{aligned}
I_x \times I_t \times I_c &\longrightarrow \mathbb{R}\\
(x, t, c) &\longmapsto u_c(x, t)
\end{aligned}
$$

Neural network u_θ

$$
\begin{aligned}
u_\theta : I_x \times I_t \times I_c &\longrightarrow \mathbb{R}\\
u_\theta(x, t, c) &\approx u_c(x, t)
\end{aligned}
$$

Now: Operator Learning

Wave equation:

$$
\begin{aligned}
\partial_t^2 u &= c^2 \partial_x^2 u && \text{on } I_x \times I_t \subset \mathbb{R}^2, \\
u &= 0 && \text{on } \partial I_x \times I_t \\
\partial_t u(\cdot, 0) &= 0 && \text{on } I_x \\
u(x, 0) &= {\color{red} f}(x) && \text{for } x \in I_x.
\end{aligned}
$$

Task: Learn u_f for **fixed** $c \in I_c$
many ${\color{red} f} : I_x \to \mathbb{R}$

Now: Operator Learning

Wave equation:

$$
\begin{aligned}
\partial_t^2 u &= c^2 \partial_x^2 u && \text{on } I_x \times I_t \subset \mathbb{R}^2, \\
u &= 0 && \text{on } \partial I_x \times I_t \\
\partial_t u(\cdot, 0) &= 0 && \text{on } I_x \\
u(x, 0) &= f(x) && \text{for } x \in I_x.
\end{aligned}
$$

Task: Learn u_f for **fixed** $c \in I_c$

many $f : I_x \to \mathbb{R}$

Operator to be learned

$$
\begin{aligned}
I_x \times I_t \times F &\longrightarrow \mathbb{R} \\
(x, t, f) &\longmapsto u_f(x, t)
\end{aligned}
$$

F set of functions, e.g. in $C(I_x, \mathbb{R})$

Now: Operator Learning

Wave equation:

$$
\begin{aligned}
\partial_t^2 u &= c^2 \partial_x^2 u && \text{on } I_x \times I_t \subset \mathbb{R}^2,\\
u &= 0 && \text{on } \partial I_x \times I_t\\
\partial_t u(\cdot, 0) &= 0 && \text{on } I_x\\
u(x, 0) &= f(x) && \text{for } x \in I_x.
\end{aligned}
$$

Task: Learn u_f for **fixed** $c \in I_c$
many $f : I_x \to \mathbb{R}$

Operator to be learned

$$
\begin{aligned}
I_x \times I_t \times F &\longrightarrow \mathbb{R}\\
(x, t, f) &\longmapsto u_f(x, t)
\end{aligned}
$$

F set of functions, e.g. in $C(I_x, \mathbb{R})$

Problem: Input of NNs must be from $\mathbb{R}^d$

Now: Operator Learning

Wave equation:

$$
\begin{aligned}
\partial_t^2 u &= c^2 \partial_x^2 u && \text{on } I_x \times I_t \subset \mathbb{R}^2, \\
u &= 0 && \text{on } \partial I_x \times I_t \\
\partial_t u(\cdot, 0) &= 0 && \text{on } I_x \\
u(x, 0) &= f(x) && \text{for } x \in I_x.
\end{aligned}
$$

Task: Learn u_f for **fixed** $c \in I_c$
many $f : I_x \to \mathbb{R}$

Operator to be learned

$$
\begin{aligned}
I_x \times I_t \times F &\longrightarrow \mathbb{R} \\
(x, t, f) &\longmapsto u_f(x, t)
\end{aligned}
$$

F set of functions, e.g. in $C(I_x, \mathbb{R})$

Problem: Input of NNs must be from $\mathbb{R}^d$

$\longrightarrow$ PINNs not applicable

Operator Learning

Why Operator Learning?

Classical Parameter-Identification
Given: Solution u of PDE
Task: Find underlying parameters f

- Iterative algorithms: Solve many similar PDEs
- Classical PDE solver like FDM or FEM: Time-consuming
- Replace by trained NN
- Less time-consuming

First Natural Idea

Wave equation:

$$
\begin{aligned}
\partial_t^2 u = c^2 \partial_x^2 u \quad & \text{on } I_x \times I_t \subset \mathbb{R}^2, \\
u = 0 \quad & \text{on } \partial I_x \times I_t \\
\partial_t u(\cdot, 0) = 0 \quad & \text{on } I_x \\
u(x, 0) = f(x) \quad & \text{for } x \in I_x.
\end{aligned}
$$

Operator to be learned

$$
\begin{aligned}
I_x \times I_t \times F &\longrightarrow \mathbb{R} \\
(x, t, f) &\longmapsto u_f(x, t)
\end{aligned}
$$

First Natural Idea

Wave equation:

$$\begin{aligned} \partial_t^2 u &= c^2 \partial_x^2 u && \text{on } I_x \times I_t \subset \mathbb{R}^2, \\ u &= 0 && \text{on } \partial I_x \times I_t \\ \partial_t u(\cdot, 0) &= 0 && \text{on } I_x \\ u(x,0) &= f(x) && \text{for } x \in I_x. \end{aligned}$$

Operator to be learned

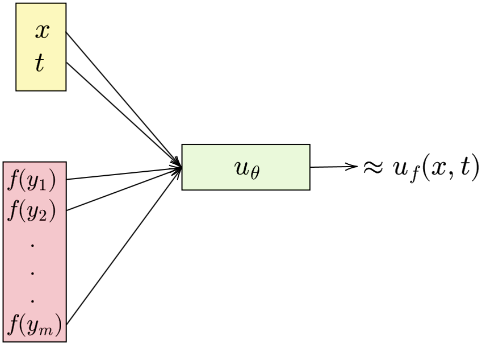
$$\begin{aligned} I_x \times I_t \times F &\longrightarrow \mathbb{R} \\ (x, t, f) &\longmapsto u_f(x, t) \end{aligned}$$

Sample f at $y_1, \dots, y_m$

$$u_\theta : I_x \times I_t \times \mathbb{R}^m \longrightarrow \mathbb{R}$$

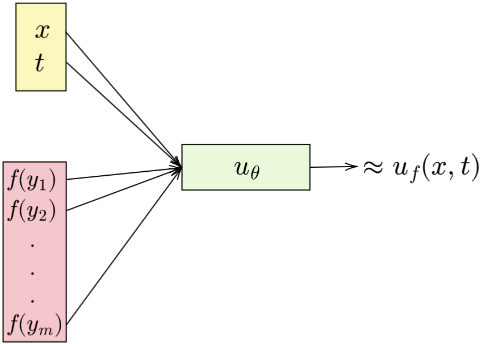
$$u_\theta\left(x, t, \begin{pmatrix} f(y_1) \\ \vdots \\ f(y_m) \end{pmatrix}\right) \approx u_f(x, t)$$

First Natural Idea



- Note: $y_1, \ldots, y_m$ are fixed
- Free evaluation of u_f at (x, t)

First Natural Idea



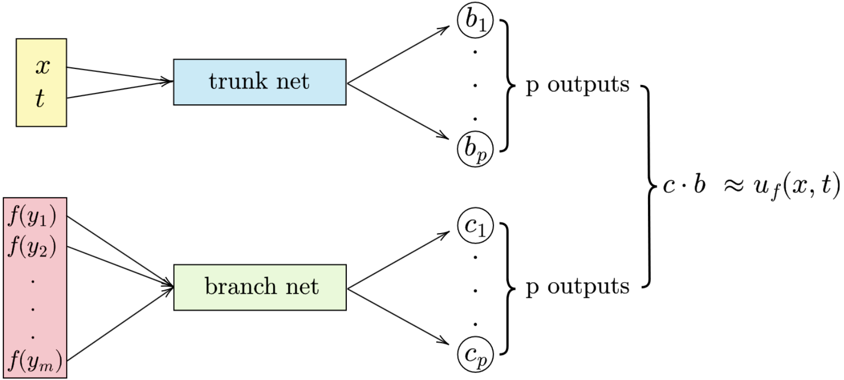
- Note: $y_1, \ldots, y_m$ are fixed
- Free evaluation of u_f at (x, t)

Problematic:

- Many $f(y_i)$-inputs versus (x, t)
  $\rightarrow$ Imbalance

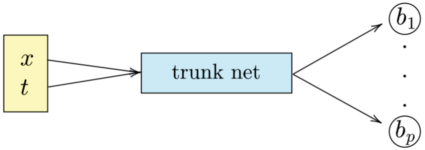
DeepONets

DeepONets[1] - Divide and Conquer



[1] Lu, Jin, Karniadakis, *Learning Nonlinear Operators (...)*, 2019

DeepONets - Interpretation



Trunk Net: $(x,t) \mapsto (b_1(x,t), ..., b_p(x,y))^T$

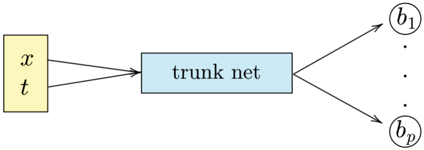
Branch Net: $(f(y_1), ..., f(y_m))^T \mapsto (c_1(f), ..., c_p(f))^T$

DeepONet: $(x,t,f(y_1), ..., f(y_m))^T \mapsto \sum_k c_k(f) b_k(x,t)$

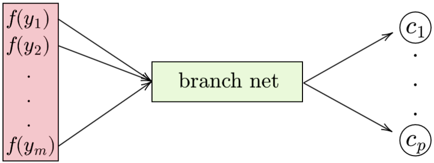
DeepONets - Interpretation

Trunk Net: $(x, t) \mapsto (b_1(x, t), ..., b_p(x, y))^T$

- Learns "basis" functions for solutions $u_f(x, t)$

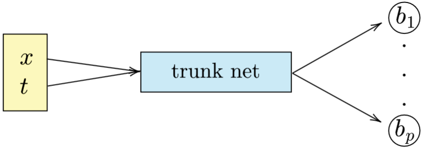


Branch Net: $(f(y_1), ..., f(y_m))^T \mapsto (c_1(f), ..., c_p(f))^T$



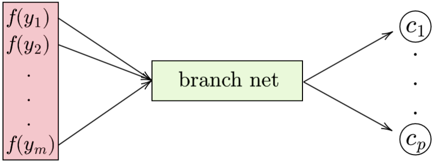
DeepONet: $(x, t, f(y_1), ..., f(y_m))^T \mapsto \sum_k c_k(f) b_k(x, t)$

DeepONets - Interpretation



Trunk Net: $(x, t) \mapsto (b_1(x, t), ..., b_p(x, y))^T$

- Learns "basis" functions for solutions $u_f(x, t)$



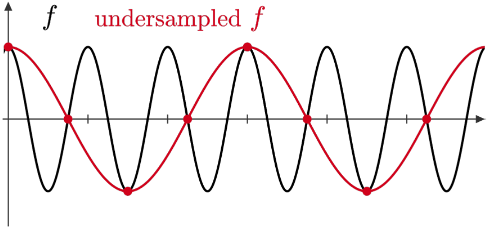
Branch Net: $(f(y_1), ..., f(y_m))^T \mapsto (c_1(f), ..., c_p(f))^T$

- Receives finite info about parameter function f
- Learns coeffients of u_f w.r.t. trunk net basis b_k

DeepONet: $(x, t, f(y_1), ..., f(y_m))^T \mapsto \sum_k c_k(f) b_k(x, t)$

Enough Info for Branch Net

- Sufficiently many sampling points $y_1, \dots, y_m$ in domain of f



- E.g. band-limited functions f
  $\to$ Sample with Shannon-Nyquist rate[2] $(2 \cdot \text{bandwidth})^{-1}$

[2] Shannon, *Communication in the Presence of Noise*, 1949

Inspired by Universal Approximation Theorem[3]

Simplified version

Theorem

Let $K \subset C([0,1]^d, \mathbb{R})$ be compact. Consider a continuous operator

$$G : [0,1]^n \times K \to \mathbb{R}.$$

For all $\varepsilon > 0$ there exist sampling points $y_1, ..., y_m \in [0,1]^d$ and a

- *branch net $\varphi_b : \mathbb{R}^m \to \mathbb{R}^p$*
- *trunk net $\varphi_t : [0,1]^n \to \mathbb{R}^p$ such that*

$$\left| G(x, f) - \langle \varphi_b\big(f(y_1), ..., f(y_m)\big), \varphi_t(x)\rangle \right| < \varepsilon \quad \text{for all} \quad f \in K, x \in [0,1]^n.$$

[3] T. Chen, H. Chen, *Universal Approximation to Nonlinear Operators (...)*, 1995

Training of DeepONets

a) Data-driven
Denotation: "DeepONet"

b) Physics-informed
Denotation: "Physics-Informed DeepONet" (PI-DeepONet)

Training of DeepONets

a) Data-driven
Denotation: "DeepONet"

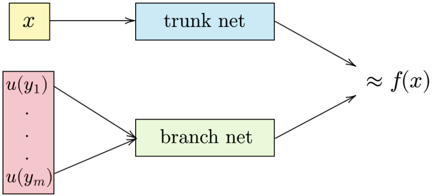
b) Physics-informed
Denotation: "Physics-Informed DeepONet" (PI-DeepONet)

TORCHPHYSICS: Provides both + combination

DeepONets for Inverse Problems

Learning the inverse Operator

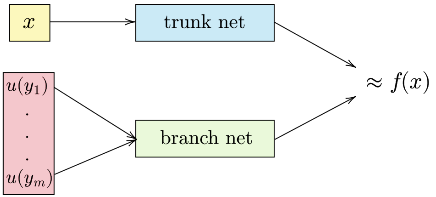
- Switch the roles of f and u
- Straightforward to implement



DeepONets for Inverse Problems

Learning the inverse Operator

- Switch the roles of f and u
- Straightforward to implement



Tikhonov scheme

- Learn forward operator, mapping (x, t, f) to $u_f(x, t)$
- Use in classical Tikhonov scheme:

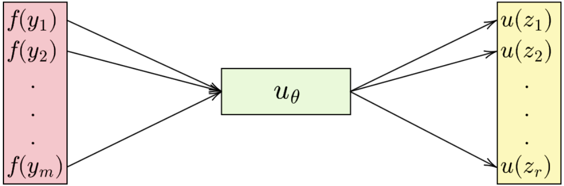
$$\min_{f \in F} \frac{1}{2} \| u_\theta(\cdot, \cdot, f) - u^\delta \|_{L^2}^2 + \frac{\alpha}{2} \| f \|_{L^2}^2$$

- More robust to noise

Alternative Operator Learning Approaches

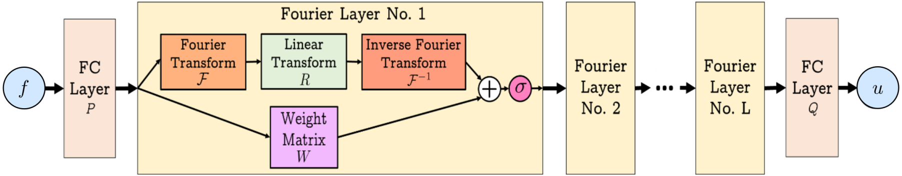
General Approach

- Sampling points $y_1, \ldots, y_m$ for functions f
- Sampling points $z_1, \ldots, z_r$ for u
- Map sampled f to sampled u



Fourier Neural Operators (FNO)[4]

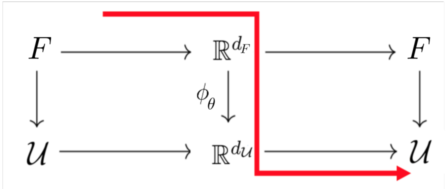
- Needs a fixed grid for both f and u
- Inspired by the convolution theorem: $(\kappa * g)(x) = \mathcal{F}^{-1}(\mathcal{F}(\kappa) \cdot \mathcal{F}(g))(x)$
- Exploits strength of Fourier transformation



[4] Li et al, *Neural operator: Graph kernel network for partial differential equations*, 2020

Model Reduction and NNs for Parametric PDEs[5]

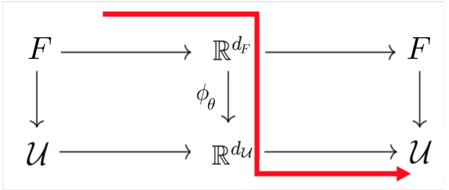
- Utilizes principal component analysis (PCA) for input f and output u
- Learn mapping $\phi_\theta : \mathbb{R}^{d_F} \to \mathbb{R}^{d_U}$



[5] Bhattacharya et al, *Model reduction and neural networks for parametric PDEs*, 2020

Model Reduction and NNs for Parametric PDEs[5]

- Utilizes principal component analysis (PCA) for input f and output u
- Learn mapping $\phi_\theta : \mathbb{R}^{d_F} \to \mathbb{R}^{d_U}$
- Usually PCA-basis only known on fixed grid
- Basis function have **not** to be learned



[5] Bhattacharya et al, *Model reduction and neural networks for parametric PDEs*, 2020

Performance of different approaches[6]

- Consider Darcy flow equation

$$
\begin{aligned}
-\nabla \cdot (f \nabla u) &= 1, \quad \text{in } (0,1)^2 \\
u &= 0, \quad \text{on } \partial(0,1)^2
\end{aligned}
$$

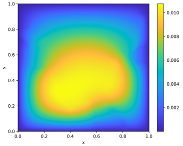
- **Forward problem:** map $f \to u$

[6] Nganyu Tanyu et al, *Deep Learning Methods for Partial Differential Equations (...)*, 2023

Performance of different approaches[6]

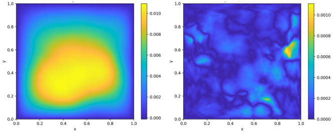
- Consider Darcy flow equation

$$
\begin{aligned}
-\nabla \cdot (f \nabla u) &= 1, \quad && \text{in } (0,1)^2 \\
u &= 0, \quad && \text{on } \partial(0,1)^2
\end{aligned}
$$

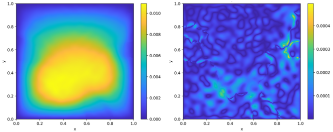


Ground Truth

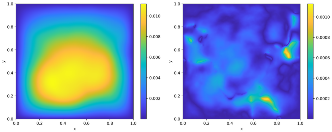
- **Forward problem:** map $f \to u$



(a) DeepONet



(b) FNO



(c) PCA

[6] Nganyu Tanyu et al, *Deep Learning Methods for Partial Differential Equations (...)*, 2023

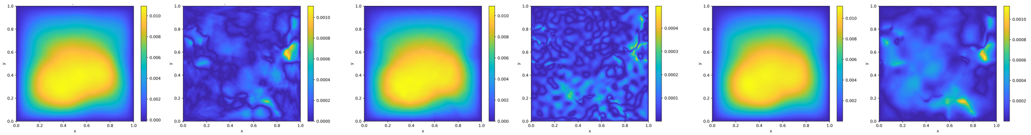
Performance of different approaches[6]

- Consider Darcy flow equation

$$
\begin{aligned}
-\nabla \cdot (f \nabla u) = 1, &\quad \text{in } (0,1)^2 \\
u = 0, &\quad \text{on } \partial(0,1)^2
\end{aligned}
$$

- **Forward problem:** map $f \to u$

	Rel. L^2 error	Evaluation time [s]
DeepONet	0.029	0.001
FNO	0.011	0.017
PCA	0.025	0.611



(a) DeepONet (b) FNO (c) PCA

[6] Nganyu Tanyu et al, *Deep Learning Methods for Partial Differential Equations (...)*, 2023

Performance of different approaches[6]

- Consider Darcy flow equation

$$\begin{aligned} -\nabla \cdot (f \nabla u) &= 1, \quad \text{in } (0,1)^2 \\ u &= 0, \quad \text{on } \partial(0,1)^2 \end{aligned}$$

- **Inverse problem:** map $u \to f$

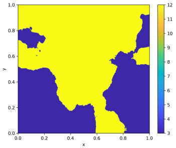
[6] Nganyu Tanyu et al, *Deep Learning Methods for Partial Differential Equations (...)*, 2023

Performance of different approaches[6]

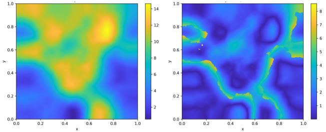
- Consider Darcy flow equation

$$
\begin{aligned}
-\nabla \cdot (f\,\nabla u) &= 1, \quad &&\text{in } (0,1)^2 \\
u &= 0, \quad &&\text{on } \partial(0,1)^2
\end{aligned}
$$

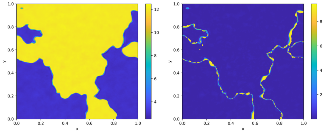
- **Inverse problem:** map $u \to f$



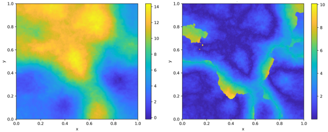
Ground Truth



(a) DeepONet



(b) FNO



(c) PCA

[6] Nganyu Tanyu et al, *Deep Learning Methods for Partial Differential Equations (...)*, 2023

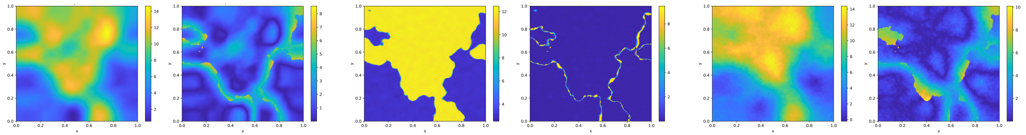
Performance of different approaches[6]

- Consider Darcy flow equation

$$-\nabla \cdot (f \nabla u) = 1, \quad \text{in } (0,1)^2$$
$$u = 0, \quad \text{on } \partial(0,1)^2$$

	Rel. L^2 error	Evaluation time [s]
DeepONet	0.222	0.001
FNO	0.149	0.016
PCA	0.099	0.154

- **Inverse problem:** map $u \to f$



(a) DeepONet (b) FNO (c) PCA

[6] Nganyu Tanyu et al, *Deep Learning Methods for Partial Differential Equations (...)*, 2023

Comparison to Other Python Libraries

Other Open-Source Python Libraries

...implementing physics-informed learning

1) DEEPXDE[7]
   - Developed by L. Lu, supervised by G. Karniadakis, Brown University, USA
   - Available on GitHub

[7] Lu et al., *DeepXDE: A Deep Learning Library for Solving Differential Equations*, 2021

Other Open-Source Python Libraries

...implementing physics-informed learning

1) DEEPXDE[7]
   - Developed by L. Lu, supervised by G. Karniadakis, Brown University, USA
   - Available on GitHub

2) NVIDIA MODULUS
   - ©2021-2022, NVIDIA Corporation
   - Available on GitHub

[7] Lu et al., *DeepXDE: A Deep Learning Library for Solving Differential Equations*, 2021

Comparison

General information

- TorchPhysics, Modulus based on PyTorch
- DeepXDE mainly on TensorFlow, also supports PyTorch, JAX, PaddlePaddle
- pip-installable
- Share similar structure/building blocks, like Domain, Conditions, etc. (different names)

Comparison

Domains and sampling

	TorchPhysics	DeepXDE	Modulus
Domain operations $\cup, \cap, \setminus$	✓ Cartesian product	✓	✓
Time-dependent domains	✓	✗	✗
STL geometry	✓	✗	✓
Grid sampling	✓	✓	✗
Random sampling	✓	✓	✓
Adaptive sampling (e.g. loss-dependent)	✓	✓	(✓) Manually

Comparison

Pre-Implemented Methods

	TorchPhysics	DeepXDE	Modulus
PINN	✓	✓ Extensions	✓
DeepRitz	✓	Manually	Manually
(PI)DeepONet	✓	✓ Extensions	✓
FNO	✗	Fourier-DeepONet	✓
PINO	✗	✗	✓