Operator Learning

Santiago Neira

Department of Mathematics and Statistics
University of Massachusetts, Amherst

January 20, 2026

Introduction

Introduction

Operator learning refers to the application of ideas from machine learning to approximate operators mapping between Banach spaces of functions. Such operators often arise from physical models expressed in terms of PDEs.

Applications:

1. Development of fast and accurate surrogate models for the solution operators of partial differential equations.
2. As a data-driven approach, operator learning techniques can be used to develop black-box simulators.
3. These operators' surrogates can also be used to complement traditional PDE solvers.

Introduction

This presentation is based on the following papers:

1. Subedi, Unique, and Ambuj Tewari. "Operator learning: A statistical perspective." arXiv preprint arXiv:2504.03503(2025).[1]
2. Kovachki, Nikola B., Samuel Lanthaler, and Andrew M. Stuart. "Operator learning: Algorithms and analysis." Handbook of Numerical Analysis 25 (2024): 419-467. [2]
3. Mora, Carlos, Amin Yousefpour, Shirin Hosseinmardi, Houman Owhadi, and Ramin Bostanabad. "Operator learning with Gaussian processes." Computer Methods in Applied Mechanics and Engineering 434 (2025): 117581. [3]

Outline

What is Operator Learning?

What is Operator Learning?

1. Let $\mathcal{U}, \mathcal{V}$ be Banach spaces of vector-valued functions

$$\mathcal{U} = \{u : \mathcal{D}_x \to \mathbb{R}^{d_i}\}, \quad \mathcal{D}_x \subseteq \mathbb{R}^{d_x}$$

$$\mathcal{V} = \{v : \mathcal{D}_y \to \mathbb{R}^{d_o}\}, \quad \mathcal{D}_y \subseteq \mathbb{R}^{d_y}$$

and let

$$\Psi^\dagger : \mathcal{U} \to \mathcal{V} \tag{1}$$

be the operator of interest, e.g. solution operator of PDE of interest.

2. The learner is provided with n iid samples $\{(u_i, \Psi^\dagger(u_i))\}_{i=1}^n$ where $u_i \sim \mu$ and μ belongs to a family of distributions $\mathcal{P}$.

What is Operator Learning?

3 Construct an estimator $\tilde{\Psi}_n^\dagger$; in practice, it is constructed within a prespecified operator class $\mathcal{F}$, e.g. bounded linear operators or neural networks.

4 For a prespecified loss function $L : \mathcal{V} \times \mathcal{V} \to \mathbb{R}_{\geqslant 0}$, the worst-case(over $\mu \in \mathcal{P}$) expected excess risk of $\tilde{\Psi}^\dagger$ is defined as

$$\mathcal{E}(\tilde{\Psi}_n^\dagger, \mathcal{F}, \mathcal{P}, \Psi^\dagger) := \sup_{\mu \in \mathcal{P}} \big(\mathbb{E}_{u_1,\ldots,u_n \sim \mu}[\mathbb{E}_{u \in \mu}[L(\tilde{\Psi}_n^\dagger(u_i), \Psi^\dagger(u))]]$$

$$- \inf_{\Psi \in \mathcal{F}} \mathbb{E}_{u \sim \mu}[L(\Psi(u), \Psi^\dagger(u))] \big)$$

Then, the learner's objective is to develop an estimation rule s.t.

$$\mathcal{E}(\tilde{\Psi}_n^\dagger, \mathcal{F}, \mathcal{P}, \Psi^\dagger) \to 0 \quad \text{as } n \to \infty$$

Loss Funtions and Estimation

Loss function

- Common choice : $L(\hat{v}, v) = ||\hat{v} - v||_{\mathcal{V}}^q$. In practice, $q = 1, 2$. Another loss function is the relative loss, given by $L(\hat{v}, v) = \frac{||\hat{v}-v||_{\mathcal{V}}^q}{||v||_{\mathcal{V}}^q}$ which is preferred in empirical setting.

Estimation

- Given n iid samples $\{(u_i, \Psi^\dagger(u_i))\}_{i=1}^n$, the estimator is learned by empirical risk minimization,

$$\Psi_n \in \arg\min_{\Psi} \frac{1}{n} \sum_{i=1}^{n} L(\Psi(u_i), \Psi^\dagger(u_i)).$$

- Optimization is performed using first-order methods (SGD and variants).
- Physics constraints (e.g., PDE structure, conservation laws) can be added as regularization terms.

Operator Classes : Encoder-Decoder

PCA-Net and DeepONet

Encoder-Decoder

Many operator learning architectures rely on extracting a finite-dimensional latent representation from infinite-dimensional spaces $\mathcal{U}$ and $\mathcal{V}$. We introduce encoder/decoder pairs

$$F_U : \mathcal{U} \to R^{d_\mathcal{U}}, \qquad G_U : \mathbb{R}^{d_\mathcal{U}} \to \mathcal{U}$$
$$F_\mathcal{V} : \mathcal{V} \to \mathbb{R}^{d_\mathcal{V}}, \qquad G_\mathcal{V} : \mathbb{R}^{d_\mathcal{V}} \to \mathcal{V}$$

such that

$$G_\mathcal{U} \circ F_U \approx I_U, \quad G_\mathcal{V} \circ F_\mathcal{V} \approx I_\mathcal{V}.$$

The operator is approximated by a map $\varphi : \mathbb{R}^{d_\mathcal{U}} \to \mathbb{R}^{d_\mathcal{V}}$ is chosen such that

$$G_\mathcal{V} \circ \varphi \circ F_\mathcal{U} \approx \Psi^\dagger \iff F_\mathcal{V} \circ \Psi^\dagger \circ G_\mathcal{U} \approx \varphi$$

Encoder-Decoder

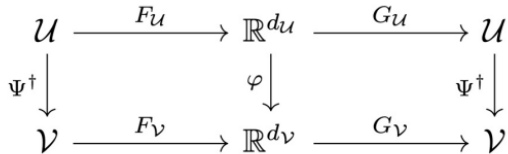


Figure 1: Latent Structure in Maps Between Banach Spaces $\mathcal{U}$ and $\mathcal{V}$

PCA-Net

In this framework, $\mathcal{U}$ and $\mathcal{V}$ are Hilbert spaces and inputs are drawn from a probability measure μ on $\mathcal{U}$.

- The encoder $F_{\mathcal{U}}$ is determined by projection onto the first $d_{\mathcal{U}}$ PCA basis functions $\{\phi_j\}_{j=1}^{d_{\mathcal{U}}}$ computed from the covariance under μ:

$$F_{\mathcal{U}} : \mathcal{U} \to \mathbb{R}^{d_{\mathcal{U}}}, \quad F_{\mathcal{U}}(u) = L(u) := (\langle \phi_j, u \rangle)_{j=1}^{d_{\mathcal{U}}}.$$

- Denoting by $\{\psi_j\}_{j=1}^{d_{\mathcal{V}}}$ the first $d_{\mathcal{V}}$ PCA basis functions under the push-forward measure $\Psi^{\dagger}_{\#}\mu$, the decoder $G_{\mathcal{V}}$ is

$$G_{\mathcal{V}} : \mathbb{R}^{d_{\mathcal{V}}} \to \mathcal{V}, \quad G_{\mathcal{V}}(\alpha) = \sum_{j=1}^{d_{\mathcal{V}}} \alpha_j \psi_j.$$

PCA-Net

- The PCA encoding and decoding on $\mathcal{U}$ and $\mathcal{V}$ are combined with a finite dimensional neural network $\alpha : \mathbb{R}^{d_{\mathcal{U}}} \times \Theta \to \mathbb{R}^{d_{\mathcal{V}}}$, $u \mapsto \alpha(u;\theta)$ where

$$\alpha(u;\theta) := (\alpha_1(u;\theta), \ldots, \alpha_{d_{\mathcal{V}}}(u;\theta)),$$

  parameterized by $\theta \in \Theta$.
- This results in an operator $\Psi_{PCA} : \mathcal{U} \to \mathcal{V}$, of the form

$$\Psi_{PCA}(u;\theta)(y) = \sum_{j=1}^{d_{\mathcal{V}}} \alpha_j(Lu;\theta)\psi_j(y), \quad \forall u \in \mathcal{U}, \quad y \in \mathcal{D}_y$$

  which defines the PCA-Net architecture.

DeepONet

The DeepONet architecture was first proposed as a practical operator learning framework in [4], building on early work by Chen and Chen [5].

- The encoder is

$$F_{\mathcal{U}} : \mathcal{U} \to \mathbb{R}^{d_{\mathcal{U}}}, \quad F_{\mathcal{U}}(u) = Lu,$$

  where L is a linear map.
  - PCA coefficients
  - Observations $\{u(x_i)\}_{i=1}^{d_{\mathcal{U}}}$ at fixed input location points $x_i \in \mathcal{D}_x$.

DeepONet

- Given a neural network (trunk-net) $\psi : \mathcal{D}_y \times \Theta_y \to \mathbb{R}^{d_{\mathcal{V}}}$, called the DeepONet decoder is defined by

$$G_{\mathcal{V}} : \mathbb{R}^{d_{\mathcal{V}}} \to \mathcal{V}, \quad G_{\mathcal{V}}(\alpha) = \sum_{j=1}^{d_{\mathcal{V}}} \alpha_j \psi_j.$$

- The above encoder and decoders on $\mathcal{U}$ and $\mathcal{V}$ are combined with a finite-dimensional neural network (branch-net) $\alpha : \mathbb{R}^{d_{\mathcal{U}}} \times \Theta_\alpha \to \mathbb{R}^{d_{\mathcal{V}}}$.
- $\Psi_{DEEP} : \mathcal{U} \to \mathcal{V}$ is defined by

$$\Psi_{DEEP}(u;\theta)(y) = \sum_{j=1}^{d_{\mathcal{V}}} \alpha_j(Lu;\theta_\alpha)\psi_j(y;\theta_\psi), \quad \forall u \in \mathcal{U}, \quad y \in \mathcal{D}_y, \quad \theta = (\theta_\alpha, \theta_\psi)$$

Example

Operator

- Antiderivative operator $\Psi^\dagger : \mathcal{U} \mapsto \mathcal{V}$ defined by the ODE $\frac{dv}{dx} = u$ with IC $v(0) = 0$.
- We use DeepONet and the library [6].

Data

- A random function u is sampled from a Gaussian random field (GRF) with the resolution $m = 100$.
- Solve v for u numerically. For each v, we have the values of $v(x)$ in the same $N_v = 100$ locations.

The training dataset has size 150 and the test dataset has size 1000. Data is organized in the following format:

Example

- Input of branch net: functions u. It is a matrix of shape (data size, m) i.e. (150,100)
- Input of trunk net: locations x of $v(x)$. It is a matrix of shape (N_v, dimension) ie. (100,1)
- Output: Values of $v(x)$ in different locations for different u. It is a matrix of shape (dataset size, N_u) e.g. (150,100) for the training dataset.

Architecture

- Branch net is a fully connected NN of size $[100, 40, 40]$. Trunk net is a fully connected NN of size $[1, 40, 40]$.
- Activation function is tanh
- Optimizer is ADAM and the loss is mean l^2 error.

Example

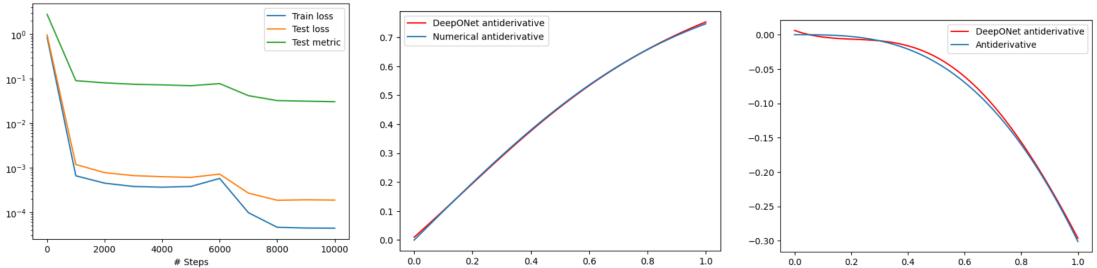


Figure 2: Figure (a) is the comparison of training loss, test loss, and mean l^2 relative error during model training. Figure (b) is the comparison between the prediction of $u(x) = e^{-x^2}$ on $[0, 1]$ using DeepONet antiderivative against a quadrature method. The l^2 error is $= 3.170e - 03$. Figure (c) is the comparison between the DeepONet prediction of $u(x) = -x\sin(x)$ on $[0, 1]$ and the antiderivative $v(x) = x\cos(x) - \sin(x)$.

Operator Classes : Neural Operators

Fourier Neural Operators

Fourier Neural Operators

The Fourier Neural Operator (FNO) architecture was introduced in [7] and [8]. It is a specific instance of a more general notion of neural operators. The general structure of such operator $\Psi_{NO} : \mathcal{U}(\mathcal{D}; \mathbb{R}^{d_i}) \to \mathcal{V}(\mathcal{D}; \mathbb{R}^{d_o})$ is

$$\Psi_{NO}(u; \theta) = \mathcal{Q} \circ \mathcal{L}_L \circ \ldots \circ \mathcal{L}_1 \circ \mathcal{R}(u), \quad \forall u \in \mathcal{U}, \tag{2}$$

where

$$u_i(x) = \mathcal{L}_i(u_{i-1})(x; \theta_i) = \sigma(W_i u_{i-1}(x) + b_i + \mathcal{K}(u_{i-1})(x; \gamma_i)),$$

- $\mathcal{R} : u(x) \mapsto Ru(x)$ (input layer), $\mathcal{Q} : u_L(x) \mapsto Qu_L(x)$ (output layer), obtained by multiplication with matrices $R \in \mathbb{R}^{d_c \times d_i}$ and $Q \in \mathbb{R}^{d_o \times d_c}$. d_c is called channel-width.

Fourier Neural Operators

- $W_i \in \mathbb{R}^{d_c \times d_c}$ and $b_i \in \mathbb{R}^{d_c \times 1}$ defines a point-wise affine transformation of the input $u_{i-1}(x) \in \mathbb{R}^{d_c \times 1}$.
- $\mathcal{K}$ is an integral operator (convolutional integral operator for FNO) defined by

$$\mathcal{K}(u)(x;\gamma) = \int_{\mathcal{D}} \kappa(x, x';\gamma) u(x') dx' \in \mathbb{R}^{d_c \times 1}$$

with $\kappa(\cdot, \cdot; \gamma_l)$ a matrix-valued integral kernel parameterized by γ.

Fourier Neural Operators

The convolutional operator can be evaluated via the Fourier transform $\mathcal{F}$, giving rise to a matrix-valued Fourier multiplier,

$$\mathcal{K}(u)(x;\gamma) = \mathcal{F}^{-1}(\mathcal{F}(\kappa(\cdot;\gamma))\mathcal{F}(u)),$$

where the Fourier transform is computed component-wise, and given by

$$\mathcal{F}(u)(k) = \int_{\mathcal{D}} u(x')e^{-ik\cdot x'}dx'.$$

Fourier Neural Operators

If we write $\kappa(x) = (\kappa_{ij}(x))_{ij=1}^{d_c}$, and $\hat{\kappa}_{k,ij}$ denotes the k-th Fourier coefficient of $\kappa_{ij}(x)$, then the i-th component $\mathcal{K}_i(u)$ of the vector-valued output function $\mathcal{K}(u)$ is given by

$$[\mathcal{K}_i(u)](x;\gamma) = \frac{1}{(2\pi)^d} \sum_{k\in\mathbb{Z}^d} \Big(\sum_{j=1}^{d_c} \hat{\kappa}_{k,ij} \mathcal{F}(u_j)(k) \Big) e^{ik\cdot x}.$$

Here, the inner sum represents the action of $\hat{\kappa} = \mathcal{F}(\kappa)$ on $\mathcal{F}(u)$, and the outer sum is the inverse Fourier transform $\mathcal{F}^{-1}$. The Fourier coefficients $\hat{\kappa}_{k,ij}$ represents the tunable parameters of the convolutional operator.

Fourier Neural Operators

- In practice, a Fourier cut-off k_{max} is introduced and the sum over k is restricted to Fourier wavenumbers $|k|_{l^\infty} \leqslant k_{\text{max}}$ with l_∞-norm, resulting in a finite number of tunable parameters $\gamma_l = \{\hat{\kappa}_{k,ij} : |k|_{l_\infty} \leqslant k_{\text{max}} : i, j = 1, \ldots, d_c.\}$
- In theory, the FNO operates directly on infinite-dimensional function spaces without using a finite latent representation. In practice, it is implemented by discretizing the input and output functions on a uniform grid. The discrete Fourier transform can be conveniently evaluated using FFT, and straightforward implementation in popular deep learning libraries is possible.

Example

Operator

- $\Psi^\dagger$ maps the initial condition u to the solution of the heat equation $\frac{\partial v}{\partial t} = \tau \nabla^2 v$ at time $T = 1$. $\tau = 0.05$ and $\mathcal{D} = [0, 1)^2$.
- We use FNO and the library NeuralOperator [9].

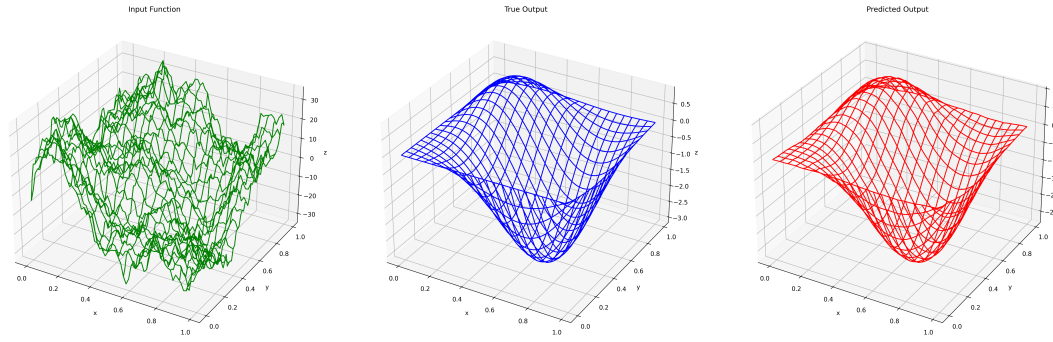
Data

- Draw 300 u initial conditions from $GP(0, (-\nabla^2 + I)^{-3/2})$. From the same distribution we draw 100 test samples.
- The solution v is obtained from finite difference method with forward Euler discretization.

Architecture

- FNO architecture uses a NN with channel width $d_c = 32$, $L = 4$ Fourier layers and GeLU activation function (GeLU $= \frac{x}{2}(1 + \text{erf}(\frac{x}{\sqrt{2}}))$.
- The optimizer is ADAM with learning rate 8e^{-3} and the l_2 loss.

Example



(a) Initial condition u (b) Numerical solution of v (c) FNO Prediction of v

Figure 3: One prediction taken from the test set. (a) is the initial condition u, (b) is the numerical solution and (c) is the prediction using FNO

Operator Learning with Gaussian Process

Gaussian Process for Operator Learning

In [3], they proposed a hybrid GP/NN-based framework for operator learning leveraging the strengths of both deep NNs and kernels for operator learning. They propose two settings, a zero-mean GP, and a NN-mean GP (FNO or DeepONet).

- Define the bounded linear observation operators

$$\phi : u \mapsto u^p := [u(x_1), \ldots, u(x_p)]^T \qquad \text{and} \qquad \psi : v \mapsto v^q := [v(y_1), \ldots, v(y_q)]^T$$

where $\{x_j\}_{j=1}^p$ and $\{y_j\}_{j=1}^q$ denote the location of the observation points in the respective domains $\mathcal{D}_x$ and $\mathcal{D}_y$.

Operator Learning Problem

Goal: Approximate the original operator $\Psi^\dagger : \mathcal{U} \to \mathcal{V}$ given $D := \{\phi(u_i), \psi(v_i))\}_{i=1}^N$ where $\{u_i, v_i\}_{i=1}^N$ are elements from $\mathcal{U} \times \mathcal{V}$.

Approach : Formulate the operator learning problem in Eq. (1) as the equivalent learning problem :

$$\tilde{\Psi}^\dagger : \mathcal{U} \times \mathcal{V}^* \to \mathbb{R},$$

where $\mathcal{V}^* = \{\gamma : \mathcal{V} \to \mathbb{R} \,|\, \gamma \text{ is continuous and linear}\}$ i.e. the dual space of $\mathcal{V}$.

Operator Learning Problem

- For any pair $(u, \varphi) \in \mathcal{U} \times \mathcal{V}^*$,

$$\tilde{\Psi}^\dagger(u, \varphi) = [\varphi, \Psi^\dagger(u)] = \int_{\mathcal{D}_y} \varphi(y)\, \Psi^\dagger(u)(y) dy.$$

- If we choose $\varphi = \delta_y$ for $y \in \mathcal{D}_y \subset \mathbb{R}^d$, then $\Psi^\dagger(u)(y) = \tilde{\Psi}^\dagger(u, \delta_y)$.
- In practice, we have a discretization of u i.e. $\phi(u) \in \mathbb{R}^p$, and we are interested in pointwise evaluations $\Psi^\dagger(u)(y)$ with $y \in \mathcal{D}_y$. Hence we consider the operator

$$\tilde{\Psi}_p^\dagger : \mathbb{R}^p \times \mathcal{D}_y \to \mathbb{R},$$

  such that $\tilde{\Psi}_p^\dagger(\phi(u), y) \approx \Psi^\dagger(u)(y)$.

Operator Learning Problem

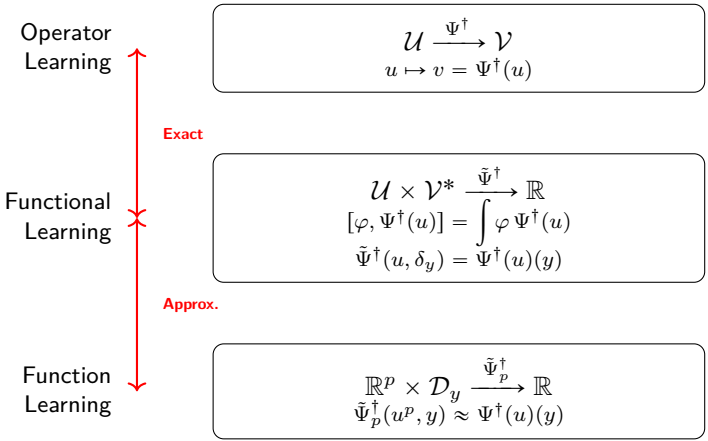


Figure 4: Diagram of the framework for operator learning. It convert the operator learning problem to a regression one which can be solved via GPs

Data-driving operator learning

To approximate $\tilde{\Psi}_p^\dagger$ we start by placing a GP prior on it:

$$\tilde{\Psi}_p^\dagger \sim GP(m(\phi(u), y; \theta), c([\phi(u), y], [\phi(u'), y'] : \beta, \sigma^2)),$$

where

- $m(\phi(u), y; \theta)$ is the prior mean function parameterized by θ.
- $c([\phi(u'), y'] : \beta, \sigma^2)$ is a kernel (Gaussian or Matern) with hyperparameters β and σ^2.

To predict $\Psi^\dagger(u^*)(y^*)$, for an unseen input function u^* at a new location y^*, we need to estimate θ, σ^2, and β using MLE.

Data-driving operator learning

Set $\mathbf{X} = \{\phi(u_i), y_j\}_{i,j=1}^{N,q}$ and $\mathbf{v} = \{[\psi(v_i)]_j\}_{i,j=1}^{N,q}$. The MLE process involves the following minimization problem

$$[\hat{\theta}, \hat{\beta}, \hat{\sigma}^2] = \arg \min_{\theta \in \Theta, \beta \in \mathcal{B}, \sigma^2 \in \mathbb{R}^+} \frac{1}{2} \log |\mathbf{C}| + \frac{1}{2} (\mathbf{v} - \mathbf{m})^T C^{-1} (\mathbf{v} - \mathbf{m}), \tag{3}$$

where

- $\mathbf{C} = c(\mathbf{X}, \mathbf{X}; \beta, \sigma^2)$ is the $Nq \times Nq$ symmetric positive-definite covariance matrix.
- $\mathbf{m} = m(\mathbf{X}; \theta)$ is the $Nq \times 1$ vector of mean function evaluations at $\mathbf{X}$
- Θ and $\mathcal{B}$ denote the search spaces for θ and β, respectively.

Data-driving operator learning

- The overall computational cost is dominated by the repeated inversion of the covariance matrix $\mathbf{C}$. (Cost : $\mathcal{O}((Nq)^3)$).
- To reduce this costs, we adopt a separable kernel of the form:

$$c([\phi(u), y], [\phi(u'), y']; \beta, \sigma^2) = c_\phi(\phi(u), \phi(u'); \beta_\phi, \sigma_\phi^2) c_y(y, y'; \beta_y, \sigma_y^2). \tag{4}$$

- WLOG, assume $\sigma_y^2 = 1$. Using this formulation the covariance matrix can be rewritten as

$$C = C_\phi \otimes C_y,$$

where

- $C_\phi = c_\phi(U, U; \beta_\phi, \sigma_\phi^2)$ with $U = \{\phi(u_1), \ldots, \phi(u_N)\}$
- $C_y = c_y(Y, Y; \beta_y)$ with $Y = \{y_1, \ldots, y_N\}$, $\beta = [\beta_\phi, \beta_y]$.

Data-driving operator learning

Using the Kronecker product we rewrite Eq (3) as

$$[\hat{\theta}, \hat{\beta}, \hat{\sigma}^2_\phi] = \arg \min_{\theta \in \Theta, \beta \in \mathcal{B}, \sigma^2_\phi \in \mathbb{R}^+} \mathcal{L}_{MLE}(\theta, \beta, \sigma^2_\phi)$$

$$= \arg \min_{\theta \in \Theta, \beta \in \mathcal{B}, \sigma^2_\phi \in \mathbb{R}^+} \log(|C_\phi|^q |C_y|^N) + (v - m)^T \text{vec}(C_y^{-1}(V - M)C_\phi^{-1}),$$

where

- V and M are formed by reshaping v and m to $q \times N$ matrices.
- vec$(\cdot)$ denotes the vectorization operation.

Data-driving operator learning

- For a GP whose mean function is parameterized with a deep NN (e.g., FNO or DeepONet), minimizing $\mathcal{L}(\theta, \beta, \sigma_\phi^2)$ can be prohibitively costly and memory intensive.
- Instead we fix β and σ_ϕ^2 to some values that ensure
  1. The covariance matrices are numerically stable.
  2. The posterior distribution regress the training data.

Now,

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \mathcal{L}_{nn}(\theta) = \arg\min_{\theta \in \Theta} (v - m)^T \mathsf{vec}(C_y^{-1}(V - M)C_\phi^{-1}), \tag{5}$$

Inference

Once the parameters of the mean and covariance functions are estimated, we obtain the predictive response for a test input function u^* at the query point y^*,

$$\eta(\phi(u^*), y^*; \hat{\theta}, \hat{\beta}, \hat{\sigma}^2_\phi) := \mathbb{E}[\tilde{\Psi}^\dagger_p(\phi(u^*), \delta_{y^*}) \,|\, \mathcal{D}] \tag{6}$$

$$= m(\phi(u^*), y^*; \hat{\theta}) + \mathsf{vec}(c_y(y^*, Y, \hat{\beta}_y) C_y^{-1} (V - M) C_\phi^{-1} c_\phi(\phi(u^*), U; \hat{\beta}_\phi, \hat{\sigma}^2_\phi)).$$

Physics-informed operator learning

Consider the general differential operator $\mathcal{T}$ acting upon $u \in \mathcal{U}$ and $v \in \mathcal{V}$ s.t $\mathcal{T} : \mathcal{U} \times \mathcal{V} \to \mathcal{O}$, where $\mathcal{O}$ is the null space, u is input function, and v denotes the unknown solution. Then, a general family of time-dependent PDEs can be expressed as:

$$\mathcal{T}(u, v) = 0 \qquad \text{in } \mathcal{D}_y \times (0, \infty),$$

$$v = f \qquad \text{on } \partial\mathcal{D}_y \times (0, \infty),$$

$$v = g \qquad \text{in } \overline{\mathcal{D}}_y \times \{0\}.$$

Given N pairs of $\{\phi(u_i), \psi(v_i)\}$ that satisfy the PDE setting, our goal is to approximate the operator $\Psi^\dagger : \mathcal{U} \to \mathcal{V}$.

Physics-informed operator learning

- Sample N_{pi} additional input functions $\{\phi(u_i)\}_{i=1}^{N_{pi}}$.
- For each $\phi(u_i)$ sample, select n_{PDE} collocation points in the domain , n_{BC} points in the boundaries and n_{IC} points at $t = 0$.

Define the losses

- $\mathcal{L}_{PDE}(\theta) = \frac{1}{N_{pi} n_{PDE}} \sum_{j=1}^{N_{pi}} \sum_{i=1}^{n_{PDE}} (\mathcal{T}(u_j, \eta(\phi(u_j), \delta_{y_i})))^2,$
- $\mathcal{L}_{BC}(\theta) = \frac{1}{N_{pi} n_{BC}} \sum_{j=1}^{N_{pi}} \sum_{i=1}^{n_{BC}} (\eta(\phi(u_j), \delta_{y_i}) - f(y_i))^2,$
- $\mathcal{L}_{IC}(\theta) = \frac{1}{N_{pi} n_{IC}} \sum_{j=1}^{N_{pi}} \sum_{i=1}^{n_{IC}} (\eta(\phi(u_j), \delta_{y_i}) - g(y_i))^2$

They quantify the errors of Eq (6) in satisfying the PDE.

Physics-informed operator learning

To estimate θ, the loss combines four components:

$$\mathcal{L}(\theta) = \mathcal{L}_{PDE}(\theta) + \alpha_{BC}\mathcal{L}_{BC}(\theta) + \alpha_{IC}\mathcal{L}_{IC}(\theta) + \alpha_{MLE}\mathcal{L}_{nn}(\theta), \tag{7}$$

- $\mathcal{L}_{PDE}$ is used as the the reference term (weight $= 1$).
- The other weights α_{BC}, α_{IC}, and α_{MLE} are adapted so that no single loss dominates.

Weights are updated using an Adam-style rule such that their gradients with respect to θ have comparable magnitudes at all optimization iterations.

$$\alpha_{BC}^{k+1} = (1-\lambda)\alpha_{BC}^{k} + \lambda \frac{\max|\nabla_\theta \mathcal{L}_{PDE}(\theta)|}{\text{mean}|\nabla_\theta \mathcal{L}_{BC}(\theta)|},$$

$\lambda = 0.9$

Fixing Parameter and Initialization

- When output observation $\psi(v)$ are dense, set β_y to a large value (e.g., 10^3).
  - For zero-mean GPs, this value is an initialization.
  - For NN-mean GPs, β_y is kept fixed.
- Set $\beta_\phi \approx 10^{-2}$, as input $\phi(u)$ typically have many location points.
  - Zero-mean GPs later fine tune β_ϕ via MLE
  - NN-mean GPs rely on the Neural Operator to adjust correlation instead of optimizing β_ϕ.
- Fix the noise variance to $\sigma_\phi^2 = 1$.

Numerical Studies

		Burgers' $N = 1000, p = 128, q = 128$		Darcy $N = 1000, p = 29^2, q = 29^2$		Advection $N = 1000, p = 40, q = 40$		Structural Mechanics $N = 1250, p = 41, q = 41^2$	
		Error	**# of Parameters**	**Error**	**# of Parameters**	**Error**	**# of Parameters**	**Error**	**# of Parameters**
DeepONet		2.15%	148,864	2.91%	715,776	0.22%	471,552	8.70%	137,856
FNO		1.93%	320,705	2.41%	6,304,257	0.66%	320,705	6.62%	888,001
GP-OR		2.15%	1280	2.75%	169,882	**2.75e−3%**	1,600	6.95%	68,921
Our GP	**0-shot Zero-mean**	2.90%	**129**	5.14%	**201**	0.17%	**41**	7.13%	**42**
	1-shot Zero-mean	2.90%	**129**	5.01%	**201**	0.17%	**41**	7.13%	**42**
	Zero-mean	2.99%	**129**	2.89%	**201**	4.11e−3%	**41**	6.74%	**42**
	DeepONet-mean	1.84%	148,864	3.44%	715,776	0.18%	471,552	7.12%	137,856
	FNO-mean	**0.08%**	320,705	**2.19%**	6,304,257	0.23%	320,705	**6.49%**	888,001

Figure 5: Summary results on four benchmark problems with $L2$ relative test error for four different operator learning methods.

Data Generation and Evaluation

Beyond the model itself, the success of operator learning depends on the choice of data distribution and evaluation framework.

Data Generation and Sampling

In applied literature $u \sim \mathrm{GP}(0, \alpha(-\nabla^2 + \beta I)^{-\gamma})$.

- Parameter γ controls the average smoothness of the generated samples. This allows to adjust γ to incorporate prior knowledge about the smoothness of input functions.
- ∇^2 captures the geometry of the domain $\mathcal{D}_x$.

Once the input function u has been generated, numerical solvers are used to obtain the corresponding solution $\Psi^\dagger(u) = v$.

Data Generation and Sampling

1. Let $\{(\varphi_j, \eta_j)\}_{j=1}^{\infty}$ be the eigenpairs of $-\nabla^2$ on $\mathcal{D}_x$.
2. By the spectral mapping theorem $\{\varphi_j\}_{j=1}^{\infty}$ are the eigenfunctions of $(-\nabla^2 + \beta I)^{-\gamma}$ with eigenvalues $\lambda_j = \alpha(\eta_j + \beta)^{-\gamma}$.
3. If $\sum_{j=1}^{\infty} \lambda_j < \infty$ then by Karhunen-Loeve decomposition decomposition $u \sim \mathsf{GP}(0, \alpha(-\nabla^2 + \beta I)^{-\gamma})$ can be decomposed as

$$u(x) = \sum_{j=1}^{\infty} \sqrt{\lambda_j} \xi_j \varphi_j(x) \qquad \forall x \in \mathcal{D}_x,$$

where $\{\xi_j\}_{j=1}^{\infty}$ are independent standard Gaussian R.V on $\mathbb{R}$.

Examples and Evaluation

Examples

- For $\mathbb{T} \simeq [0, 1]$, the eigenfunctions are Fourier basis, with eigenvalues $\lambda_j = \alpha(\beta + 4\pi^2|j|^2)^{-\gamma}$.
- Matérn or radial basis function (RBF) can also be used. The eigenfunctions of the RBF kernel are the Hermite polynomials with eigenvalues that decay exponentially fast.

Evaluation and Out-of-Distribution Generalization

- If the training distribution μ is a Gaussian process with covariance $\alpha(-\nabla^2 + \beta I)^{-\gamma_1}$, the test distribution μ_{test} could have covariance of the form $\alpha(-\nabla^2 + \beta I)^{-\gamma_2}$ for some $\gamma_2 < \gamma_1$.

Universal Approximation

Universal Approximation

The methods discussed so far aim to approximate operators acting between infinite-dimensional function spaces. A natural theoretical question is whether such approximation is even possible. Universal approximation theory addresses this by identifying the classes of operators for which these learning architectures can approximate any target operator to arbitrary accuracy.

Universal Approximation

Theorem (DeepONet Universality)

Suppose that $\sigma \in C(\mathbb{R})$ is a non-polynomial activation function. Let $\mathcal{D} \subset \mathbb{R}^d$ be a compact domain with Lipschitz boundary. Let $\Psi^\dagger : C(\mathcal{D}) \to C(\mathcal{D})$ be a continuous operator. Fix a compact set $K \subset C(\mathcal{D})$. Then for any $\epsilon > 0$, there are positive integers $d_{\mathcal{U}}$, $d_{\mathcal{V}}, N$, sensor points $x_1, \ldots, x_{d_{\mathcal{U}}} \in \mathcal{D}$, and coefficients $c_i^k, \xi_{ij}^k, b_i, w_k, \zeta_k$ with $i = 1, \ldots, N, j = 1, \ldots, d_{\mathcal{U}}, k = 1, \ldots, d_{\mathcal{V}}$, such that

$$\sup_{u \in K} \sup_{x \in \mathcal{D}} |\Psi^\dagger(u)(x) - \sum_{k=1}^{d_{\mathcal{V}}} \sum_{i=1}^{N} c_i^k \sigma\big(\sum_{j=1}^{d_{\mathcal{U}}} \xi_{ij}^k u(x_j) + b_i \big) \sigma(w_k x + \zeta_k)| \leqslant \epsilon. \tag{8}$$

Universal Approximation

Take the linear encoder $Lu = (u(x_1), \ldots, u(n_{d_{\mathcal{U}}}))$, the shallow branch-net α, with components

$$a_k(Lu) = \sum_{i=1}^{N} c_i^k \sigma(\sum_{j=1}^{d_{\mathcal{U}}} \xi_{ij}^k u(x_j) + b_i),$$

and trunk-net ψ, with components

$$\psi_k(y) = \sigma(w_k x + \zeta_k).$$

With these definitions, inequality (8) can be written in the equivalent form,

$$\sup_{u \in K} ||\Psi^\dagger(u) - \sum_{k=1}^{d_{\mathcal{V}}} \alpha_k(Lu)\psi_k||_{C(\mathcal{D})} \leqslant \epsilon.$$

Universal Approximation PCA-Net

Theorem (PCA-Net universality)

Let $\mathcal{U}$ and $\mathcal{V}$ be separable Hilbert spaces and let $\mu \in \mathcal{P}(\mathcal{U})$ be a probability measure on $\mathcal{U}$. Let $\Psi^\dagger : \mathcal{U} \to \mathcal{V}$ be a μ-measurable mapping. Assume the following moment conditions,

$$\mathbb{E}_{u\sim\mu}[||u||_{\mathcal{U}}^2], \mathbb{E}_{u\sim\mu}[||\Psi^\dagger(u)||_{\mathcal{V}}^2] < \infty.$$

Then for any $\epsilon > 0$, there are dimension $d_{\mathcal{U}}, d_{\mathcal{V}}$, a require amount of data N, a neural network ψ depending on this data, such that the PCA-Net, $\Psi = G_{\mathcal{V}} \circ \psi \circ F_{\mathcal{U}}$, satisfies

$$E_{\{u_j\}\sim\mu^{\otimes N}}[\mathbb{E}_{u\sim\mu}[||\Psi^\dagger(u) - \Psi(u; \{u_j\}_{j=1}^N)||_{\mathcal{V}}^2]] < \epsilon,$$

where the outer expectation is with respect to the iid data samples $u_1, \ldots, u_N \sim \mu$, which determine the empirical PCA encoder and reconstruction.

Averaging Neural Operator (ANO)

The universality of FNO was established in [10], using ideas from Fourier analysis. However, since many neural operator architectures differ mainly in how their kernel is parameterized, it is desirable to have a universality result that applies to a wide range of choice for the integral kernel. The Averaging Neural Operator (ANO) provides such a general framework. It captures a wide family of kernel-based neural operators, including:

- Wavelet Neural Operator
- Laplace Neural Operator
- Low-Rank Neural Operator

Averaging Neural Operator

- Up to non-essential details, the ANO introduced in [11] is a composition of nonlinear layers of the form,

$$\mathcal{L}(v;\gamma_l)(x) = \sigma\big(W_l v(x) + b_l(x) + V_l \int_{\mathcal{D}} v(y)dy\big), \quad (l = 1, \ldots, L),$$

  where $W_l, V_l \in \mathbb{R}^{d_c \times d_c}$ are matrices, and $b_l(x)$ is a bias functions.

- The resulting ANO is an operator of the form,

$$\Psi(u;\theta) = \mathcal{Q} \circ \mathcal{L}_L \circ \ldots \circ \mathcal{L}_1 \circ \mathcal{R}(u),$$

  where $\mathcal{R}$ and $\mathcal{Q}$ are the linear input and output layers.

Universal Approximation : ANO

Theorem

Suppose that $\sigma \in C(\mathbb{R})$ is a non-polynomial activation function. Let $\mathcal{D} \subset \mathbb{R}^d$ be a compact domain with Lipschitz boundary. Let $\Psi^\dagger : C(\mathcal{D}) \to C(\mathcal{D})$ be a continuous operator. Fix a compact set $K \subset C(\mathcal{D})$. Then for any $\epsilon > 0$, there exists an ANO $\Psi : C(\mathcal{D}) \to C(\mathcal{D})$ such that

$$\sup_{u \in K} ||\Psi^\dagger(u) - \Psi(u)||_{C(\mathcal{D})} < \epsilon.$$

Quantitative Error and Complexity Estimates

Quantitative Error and Complexity Estimates

Universality of neural operator architectures is an important a necessary condition for their success, but it is purely qualitative and does not explain the efficiency of these methods in practice. To understand their practical performance, we need a quantitative theory of operator learning: one that provides explicit error bounds and complexity estimates. In particular, we seek to answer questions such as: *for a given accuracy* $\epsilon > 0$, *how large must the model be, or how many samples are needed, to achieve this accuracy?*

- In the following slides, we consider Lipschitz operator. [2] also review complexity estimates for linear and holomorphic operators.

Error Decomposition

Each part of this decomposition represent a source of error and the total encoder-decoder-net approximation error $\mathcal{E}$ is bounded by three contributions

$$\mathcal{E} \lesssim \mathcal{E}_{\mathcal{U}} + \mathcal{E}_{\mathcal{V}} + \mathcal{E}_{\psi},$$

where

$$\mathcal{E}_{\mathcal{U}} = \sup_{u} ||u - G_{\mathcal{U}} \circ F_{\mathcal{U}}(u)||_{\mathcal{U}} \quad \text{(encoding error)}$$

$$\mathcal{E}_{\mathcal{V}} = \sup_{v} ||v - G_{\mathcal{V}} \circ F_{\mathcal{V}}(v)||_{\mathcal{V}} \quad \text{(reconstruction error)}$$

$$\mathcal{E}_{\psi} = \sup_{\alpha} ||F_{\mathcal{V}} \circ \Psi^{\dagger} \circ G_{\mathcal{U}}(\alpha) - \psi(\alpha)||_{\mathcal{V}} \quad \text{(neural network approx error)}$$

Neural Network Approximation Error

Theorem

A function $f \in W^{k,\infty}([0,1]^d)$ can be approximated to uniform accuracy $\epsilon > 0$,

$$\sup_{x \in [0,1]^d} |f(x) - \psi(x)| \leqslant \epsilon,$$

by a ReLU neural network ψ with at most $\mathcal{O}(\epsilon^{-d/k} \log(\epsilon^{-1}))$ tunable parameters.

Combined Error Analysis for Encoder-Decoder-Nets.

- Under reasonable assumptions on the input functions, the encoding error can often be shown to decay at an algebraic rate in the $d_{\mathcal{U}}$, e.g.

$$\mathcal{E}_{\mathcal{U}} \lesssim d_{\mathcal{U}}^{-\alpha}. \tag{9}$$

- For example, if we assume that the input functions are defined on a bounded domain $\mathcal{D} \subset \mathbb{R}^d$ and subject to a smoothness constraint such as a uniforms bound on their k-th derivative, then a decay rate $\alpha = k/d$ can be achieved (depending on the precise setting).

Combined Error Analysis for Encoder-Decoder-Nets.

- Under similar assumptions on the set of output functions, depending on the properties of the underlying operator $\Psi^\dagger$, the reconstruction error on the output space often also decays algebraically,

$$\mathcal{E}_{\mathcal{V}} \lesssim d_{\mathcal{V}}^{-\beta}, \tag{10}$$

  where the decay rate β can e.g. be estimated in terms of the smoothness of the output functions under $\Psi^\dagger$.

Combined Error Analysis for Encoder-Decoder-Nets.

- Given latent dimensions $d_{\mathcal{U}}$ and $d_{\mathcal{V}}$, the size of the neural networks ψ that is required to approximate the encoder operator mapping $\varphi : \mathbb{R}^{d_{\mathcal{U}}} \to \mathbb{R}^{d_{\mathcal{V}}}$ with approximation error bound,

$$\mathcal{E}_{\psi} \leqslant \epsilon,$$

roughly scales as

$$\text{size}(\psi) \sim d_{\mathcal{V}} \epsilon^{-d_{\mathcal{U}}/k}, \tag{11}$$

when the only information on the underlying operator is captured by its degree of smoothness k.

Combined Error Analysis for Encoder-Decoder-Nets.

- Given the error decomposition, we require each error contribution to be bounded by ϵ. In view of (9) and (10), this can be achieved provided that $d_{\mathcal{U}} \sim \epsilon^{-1/\alpha}, d_{\mathcal{V}} \sim \epsilon^{-1/\beta}$. Assuming this errors, we arrive at a neural network size of roughly of the form,

$$\text{size}(\psi) \sim \epsilon^{-1/\beta} \epsilon^{-c\epsilon^{-1/\alpha}/k}.$$

- In particular, we note the exponential dependence on ϵ^{-1}, resulting in a size estimate,

$$\text{size}(\psi) \gtrsim \exp(\frac{c\epsilon^{-1/\alpha}}{k}).$$

Combined Error Analysis for Encoder-Decoder-Nets.

- This super-algebraic dependence of the complexity on ϵ^{-1} has been termed the "curse of dimensionality" or "curse of parametric complexity".
- The curse of parametric complexity can be viewed as an infinite-dimensional scaling limit of the finite-dimensional curse of dimensional, represented by the $d_{\mathcal{U}}$-dependency of the bound $\epsilon^{-d_{\mathcal{U}}/k}$, and arises from the finite-dimensional CoD by observing that the required latent dimensional $d_{\mathcal{U}}$ itself depends on ϵ, with scaling $d_{\mathcal{U}} \sim \epsilon^{-1/\alpha}$.

Sample Complexity

Assume

- Lipschitz operator $\Psi^\dagger : \mathcal{U} \to \mathcal{V}$, where $\mathcal{U}, \mathcal{V} = L^2([-1,1]^d)$.
- There exists $C > 0$, s.t the probability measure $\mu \in \mathcal{P}(\mathcal{U})$ and its push-forward $\Psi^\dagger_{\#}(\mu) \in \mathcal{P}(\mathcal{V})$ are supported on

$$K := \{u \in: L^2([-1,1]^d) \,|\, u \text{ is periodic, } ||u||_{C^{k,\alpha}} \leqslant C\}.$$

To ensure the total error $\mathcal{E} \leqslant \epsilon$, first choose $d_{\mathcal{U}}, d_{\mathcal{V}} \sim \epsilon^{-d/s}$, consistent with previous discussion. We choose a number of samples of roughly the size

$$N \sim \epsilon^{-(2+d_{\mathcal{U}})/2}.$$

- The additional ϵ-dependency of $d_{\mathcal{U}}$ implies that

$$N \gtrsim \exp(c\epsilon^{-d/s}),$$

exhibits and exponential curse of dimensionality.

Conclusions

1. Operator learning provides a unified framework for learning mappings between function spaces. Architectures such as DeepONet, PCA-Net, and FNO achieve strong empirical performance by exploiting structure such as low-dimensional latent representations.
2. Many neural operator architectures satisfy universal approximation properties. Such Universal approximation is a necessary condition but not a sufficient one for the complete success of this theory.

Conclusions

3 For general Lipschitz operators, standard architectures require a number of parameters that grows exponentially in ϵ^{-1}, raising doubts about whether operator learning at this level of generality is possible. In contrast, holomorphic operators admit algebraic complexity in both model size and sample complexity.

4 Gaussian Process operator learning provides competitive results. Both zero-mean and NN-mean GP frameworks achieve accuracy comparable to DeepONet and FNO. NN-mean GP leverages the strengths of both kernels and NNs, while the zero-mean GP already performs well without training and naturally provides uncertainty quantification.

References I

Unique Subedi and Ambuj Tewari.
Operator learning: A statistical perspective, 2025.

Nikola B. Kovachki, Samuel Lanthaler, and Andrew M. Stuart.
Chapter 9 - operator learning: Algorithms and analysis.
In Siddhartha Mishra and Alex Townsend, editors, *Numerical Analysis Meets Machine Learning*, volume 25 of *Handbook of Numerical Analysis*, pages 419–467. Elsevier, 2024.

Carlos Mora, Amin Yousefpour, Shirin Hosseinmardi, Houman Owhadi, and Ramin Bostanabad.
Operator learning with gaussian processes.
Computer Methods in Applied Mechanics and Engineering, 434:117581, 2025.

References II


Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Karniadakis.
Learning nonlinear operators via deeponet based on the universal approximation theorem of operators.
Nature Machine Intelligence, 3:218–229, 03 2021.


Tianping Chen and Hong Chen.
Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems.
IEEE Transactions on Neural Networks, 6(4):911–917, 1995.


Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis.
DeepXDE: A deep learning library for solving differential equations.
SIAM Review, 63(1):208–228, 2021.

References III


Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar.
Fourier neural operator for parametric partial differential equations.
arXiv preprint arXiv:2010.08895, 2020.


Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar.
Neural operator: Learning maps between function spaces with applications to pdes.
Journal of Machine Learning Research, 24(89):1–97, 2023.


Jean Kossaifi, Nikola Kovachki, Zongyi Li, David Pitt, Miguel Liu-Schiaffini, Valentin Duruisseaux, Robert Joseph George, Boris Bonev, Kamyar Azizzadenesheli, Julius Berner, and Anima Anandkumar.
A library for learning neural operators.
arXiv preprint arXiv:2412.10354, 2025.

References IV


Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra.
On universal approximation and error bounds for fourier neural operators.
Journal of Machine Learning Research, 22(290):1–76, 2021.


Samuel Lanthaler, Zongyi Li, and Andrew M. Stuart.
Nonlocality and nonlinearity implies universality in operator learning.
Constructive Approximation, 62(2):261–303, Oct 2025.