Scientific Machine Learning Workshop

Lecture 6: Operator Learning

Ulisses Braga Neto

Department of Electrical and Computer Engineering
Scientific Machine Learning Lab (SciML Lab)
Texas A&M Institute of Data Science (TAMIDS)
Texas A&M University

Cenpes
August 2025

Introduction

- In operator learning, one is not interested in merely obtaining a single solution for a single PDE, but to learn a PDE operator.
- An operator is a mapping between general vector spaces, which are typically infinite-dimensional (spaces of functions).
- For example, the mapping can be between initial condition and the evolved solution at a later time, or between parameter and the solution.
- The parameter could be a vector (finite-dimensional) or a coefficient function (infinite-dimensional). An example of the latter is a space-varying diffusion coefficient in the Heat PDE.
- Most operator learning schemes are based on deep neural networks.

Operator Learning from Coefficient Function

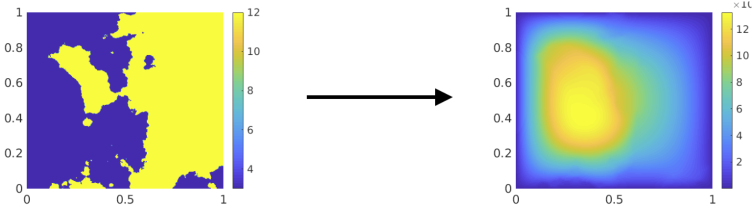
- Consider Darcy's Equation:

$$-\nabla \cdot (a(x,y)\nabla u(x,y)) = f(x,y)\,, \quad (x,y) \in \Omega\,,$$
$$u(x,y) = 0\,, \quad (x,y) \in \partial\Omega\,,$$

  where $u(x,y)$ is the fluid pressure and $a(x,y)$ is a space-varying permeability function in a porous medium.

- Given a new porosity profile, the trained model produces an approximation to the corresponding fluid pressure.

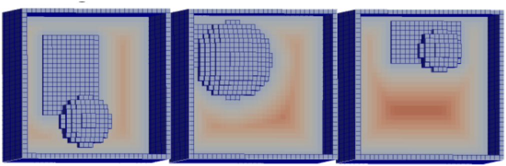
$a(x,y)$ $\qquad\qquad\qquad\qquad$ $u(x,y)$

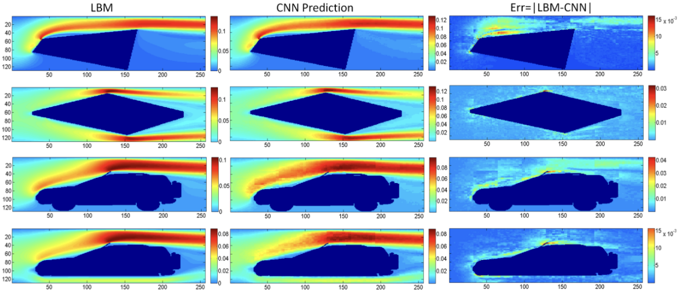


(Plots from: Kovachki, Lanthaler, and Stewart, "Operator Learning: Algorithms and Analysis", ArXiV, 2024.)

Operator Learning from Geometry

- One can use a signed distance function as input to learn from geometry.





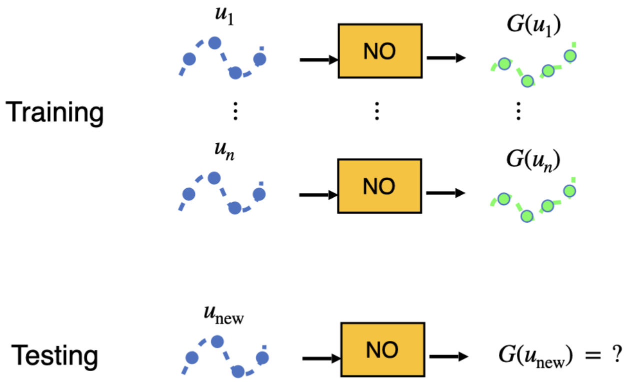
(Plots from: Guo, Li, and Iorio, "Convolutional Neural Networks for Steady Flow Approximation", KDD, 2016.)

Many-Query Problems

- The justification of operator learning is that in many applications, a PDE has to be solved several times for different initial conditions, boundary conditions, or coefficient functions.
- This is the case for example in inverse problems, uncertainty quantification, and product design.
- *Amortized inference* is the paradigm where training is time-consuming (collecting experimental or simulated data, training a large neural network), but test time is short.
- Hence, the large training time is amortized by applying the trained model many times, without further training.

Data-Driven Paradigm

- The goal is to learn the mapping from input to output from n training pairs and then predict the output for a new input.
- *"The physics is in the data."* This may be an advantage over physics-informed ML if there is missing/approximated physics.

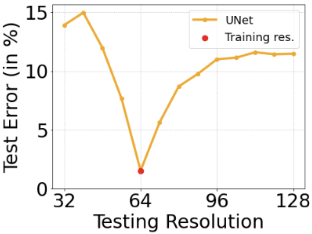


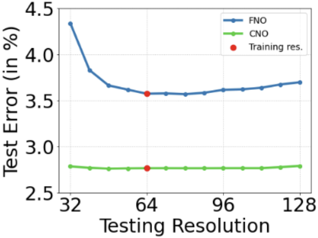
The Issue of Discretization

- In practice, the input function must be discretized for use in a computer algorithm. This discretization can be, for example:
  - Sampling the function over a uniform grid.
  - Using a parametric representation.
  - Using a (finite) basis decomposition.

- One would like to have some form of *discretization invariance*.

- Comparing the test errors at different resolutions reveals the issue.





(Plots from: Bartolucci et al., "Are Neural Operators Really Neural Operators?", Research Report, 2023.)

Mathematical Definition

- Let X be a vector space over the reals. A *norm* on X is a function $||\cdot|| : X \to R$ such that, for every $f, g \in X$,
  1. $||f|| > 0$, if $f \neq 0$.
  2. $||cf|| = |c|||f||$ for every $c \in R$ (in particular, $||0|| = 0$).
  3. $||f + g|| \leq ||f|| + ||g||$ (Minkowski's Inequality).

- A Cauchy sequence $f_1, f_2, \ldots \in X$ is such that, given any $\varepsilon > 0$, there is a N such that

$$||f_i - f_j|| < \varepsilon\,, \quad \text{for all } i, j > N\,.$$

- Space X is *complete* if all Cauchy sequences have a limit in X.

- Space X is a *Banach Space* if it is normed and complete.

Mathematical Definition

- Example: The space $C(K)$ of all reall-valued continuous functions defined on a compact (closed and bounded) set $K \subset R^d$ is a Banach space, with the supremum norm:

  $$||f|| = \max_{x \in K} |f(x)| .$$

  Note that $||f||$ is always well-defined and finite due to the fact that continuous functions map compacts into compacts.

- If the norm of a Banach space H arises from an inner product

  $$||f|| = \sqrt{\langle f, f \rangle} ,$$

  then H is a *Hilbert space* (see the definition in Lecture 6).

- RKHS and the L^2 space (all defined in Lecture 6) are Hilbert (and thus Banach) spaces. However, the Banach space $C(K)$ defined above is not a Hilbert space (the supremum norm does not arise from an inner product).

Mathematical Definition

- An *operator* $G : X \to U$ is a mapping between Banach spaces.
- The operator $G : X \to U$ is continuous if, for every $\varepsilon > 0$, there is a $\delta > 0$ such that, for every $f_1, f_2 \in X$,

$$||f_1 - f_2||_X < \delta \;\Rightarrow\; ||G(a_1) - G(a_2)||_U < \varepsilon\,.$$

- The operator G is linear if $G(c_1 f_1 + c_2 f_2) = c_1 G(f_1) + c_2 G(f_2)$, for every $f_1, f_2 \in X$, otherwise, it is nonlinear.
- The operator G is bounded if the *operator norm* $||G||$ is finite:

$$||G|| := \sup_{f \neq 0} \frac{||G(f)||}{||f||} = \sup_{||f||=1} ||G(f)|| < \infty\,.$$

- Theorem: a linear operator G is continuous if and only if it is bounded. (There are discontinuous linear operators!)

Mathematical Definition

- Consider a continuous (linear or nonlinear) groundtruth operator $G : X \to U$ between Banach spaces X and U, and let μ be a probability distribution over X.

- Consider a parametrized operator $G_\theta : X \times U$, where $\theta \in \Theta$ is the parameter. We would like to find θ such that $G_\theta \approx G$, in the sense that $E_{f\sim\mu}\left[||G_\theta(f) - G(f)||\right]$ is minimized.

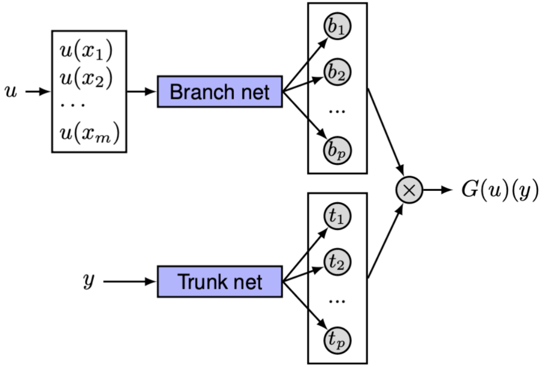
- In practice, this parametric operator is to be learned from sample i.i.d. pairs $\{(f_1, u_1), \dots, (f_n, u_n)\}$ where $f_i \sim \mu$ and $u_i = G(f_i)$ (possibly contaminated by noise), by minimizing an empirical loss

$$\mathcal{L}(\theta) = \frac{1}{nk} \sum_{i=1}^{n} \sum_{j=1}^{k} |G_\theta(f_i)(x_j) - u_i(x_j)|^2$$

where $x_1, \dots, x_k$ are test points.

DeepONets

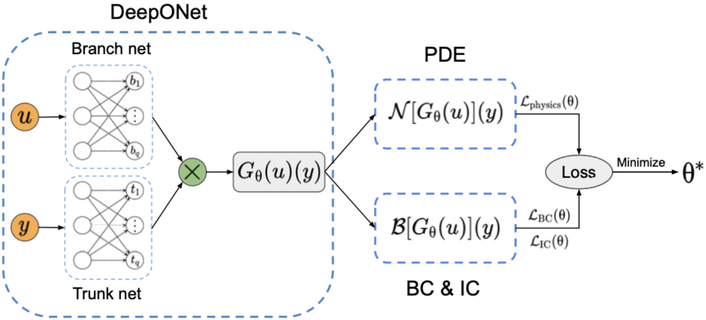
- Deep Operator Networks (DeepONets) is a popular architecture for implementing G_θ.
- Notice the discretization points $x_1, \dots, x_m$.



(Diagram from: Lu, Jin, and Karniadakis, "DeepONets", ArXiV, 2020.)

Physics-Informed DeepONets

- It is possible to add physics biases directly, using the same method as used in PINNs: a soft penalty via a loss term.



(Diagram from: Wang, Wang, and Perdikaris, "Physics-Informed DeepONets", *Science Advances*, 2021.)

Universal Representation Theorem for DeepONets

- The following theorem extends a result by Chen and Chen (1995) for shallow neural networks.

- **Theorem.** Let K_1 and K_2 be compact sets in R^p and R^d, respectively, and V be a compact set in $C(K_1)$. If $G : V \to C(K_2)$ is a nonlinear continuous operator, then, for any $\varepsilon > 0$, there exist continuous vector functions $\mathbf{f} : R^d \to R^p$, $\mathbf{g} : R^m \to R^p$, and $x_1, \ldots, x_m \in K_1$, such that

$$\left| G(u)(y) - \underbrace{\mathbf{g}(u(x_1), \ldots, u(x_m))}_{\text{branch}} \cdot \underbrace{\mathbf{f}(x_{\text{query}})}_{\text{trunk}} \right| < \varepsilon \,,$$

  for all $u \in V$ and $x_{\text{query}} \in K_2$, where "$\cdot$" is the dot product in R^p.

- The functions $\mathbf{f}$ and $\mathbf{g}$ can be approximated by diverse classes of neural networks.

Neural Operators

- A distinct class of architectures are based on the idea of extending a neural network by replacing matrix operations by linear operators.

- Matrix multiplication can be extended to infinite-dimensional spaces by means of a *kernel integral operator*:

$$K_\theta(f)(x) = \int k_\theta(x, y) f(y)\, dy$$

  where $k_\theta(x, y)$ is a parametrized kernel function.

- The output $v_{k+1}$of layer $k+1$ in the operator network is computed in terms of the previous layer as

$$v_{k+1}(x) = \sigma\left(W(v_k)(x) + \int k_\theta(x, y) v_k(y)\, dy \right),$$

  where σ is a nonlinearity and W is a linear operator (representing the "bias" term).

Fourier Neural Operators

- Fourier Neural Operators (FNO) assume a translation-invariant kernel $k_\theta(x, y) = k_\theta(x - y)$, in which case the kernel integral operator becomes a convolution operator:

$$K_\theta(f)(x) = \int k_\theta(x - y) f(y)\, dy = (k_\theta \star f)(x)$$

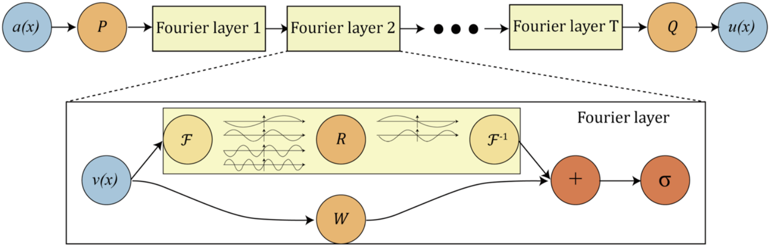
- The convolution is computed in the frequency domain using the Fourier transform:

$$K_\theta(f)(x) = \mathcal{F}^{-1}(\mathcal{F}(k_\theta) \cdot \mathcal{F}(f))(x)\,.$$

- The input function is discretized, so that the convolution is computed using the Fast Fourier Transform (FFT), which is fast.

Fourier Neural Operators

- FNO also includes lifiting P and projection Q layers.
- The entire architecture can be represented as follows.



(Diagram from: Li et al., "Fourier Neural Operators for Parametric Partial Differential Equations", *ICLR*, 2021.)

Foundation Models

- A more recent development is foundation models for scientific problems.
- This is inspired by the use of LLM as foundation models.
- A foundation model can be trained on a large body of data and be adapted for new tasks, perhaps with no further training.
- In SciML, foundation models correspond to *multiple operator learning*.
- The associated well-posedness question can be resolved by
  - *Fine-tuning* on a small amount of data from the new task.
  - By using *in-context learning*.

In-Context Learning

- In-context Learning (ICL) is an increasingly popular form of machine learning, where a trained neural network learns from examples and a query provided by a user prompt.
- ICL is an enabling technology for foundation models. It is a form of *meta-learning*, or learning to learn. Given a few examples and a prompt, the model can learn the correct response, without further training. It is also a form of *few-shot learning*.
- For example, one might want to use ChatGPT to predict whether a restaurant review is positive or negative, but the model wasn't trained on restaurant reviews. So examples are given in a prompt.

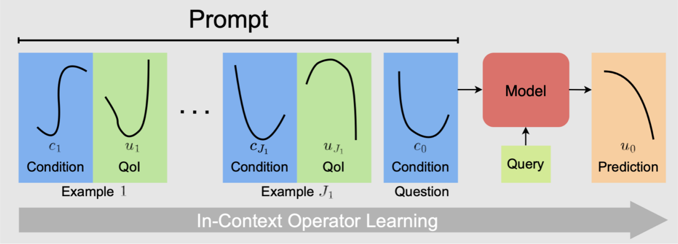
prompt: "Delicious food!" → positive, "The food is awful." → negative.

query: "Good meal!" **prediction**: positive.

(Example from: Dong et al., "A Survey on In-context Learning", ArXiV, 2024.)

In-Context Operator Learning

- In ICL for PDEs, the in-context examples consist of function pairs (e.g. coefficient function and solution).
- The query is a new input function that may not have been seen during training, and the goal is to predict the corresponding output function
- In regular ML, this creates a well-posedness question, requiring fine-tuning. Not so in ICL, since the operator can be disambiguated from the prompt.

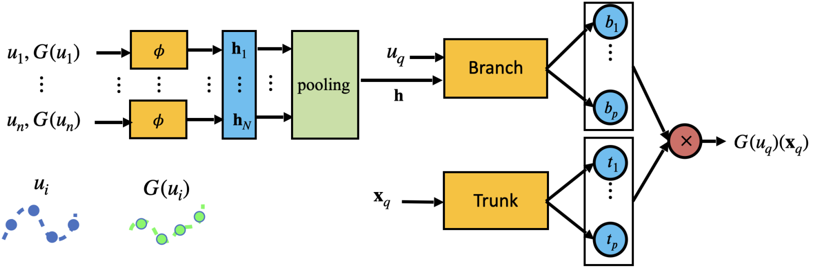


(Diagram from: Yang et al., "In-context operator learning for differential equation problems", *PNAS*, 2023.)

DeepOSets for In-Context Operator Learning

- DeepOSets (= DeepSets + DeepONets) is a new architecture for in-context operator learning proposed by our group that includes a permutation-invariance bias in the prompt.

- It processes the prompt in parallel and has complexity O(n) in the number of examples, and O(1) after the first query, which makes it more efficient and faster than tranformer-based alternatives.



(Diagram from: Chiu, Wong, and Braga-Neto, "DeepOSets", Neurips'2024 Workshop FM4Science.)

Example: Poisson Boundary Value Problem

- Consider the PDE:

$$\frac{d^2 u(x)}{dx^2} = f(x), \quad 0 < x < 1,$$
$$u(0) = u_0, \;\; u(1) = u_1 .$$

- Forward problem: Given the forcing function $f(x)$, find the solution $u(x)$.

- Inverse problem: Given the solution $u(x)$, find the forcing function $f(x)$ (control problem).

- Very importantly, the boundary values u_0 and u_1 are unknown and must be learned in-context.

Example: Reaction-Diffusion Boundary Value Problem

- Consider the PDE:

$$a\frac{d^2u(x)}{dx^2} + k(x)u(x) = c\,, \quad 0 < x < 1\,,$$
$$u(0) = u_0\,,\ \ u(1) = u_1\,.$$

- Forward problem: Given the coefficient function $k(x)$, find the solution $u(x)$.

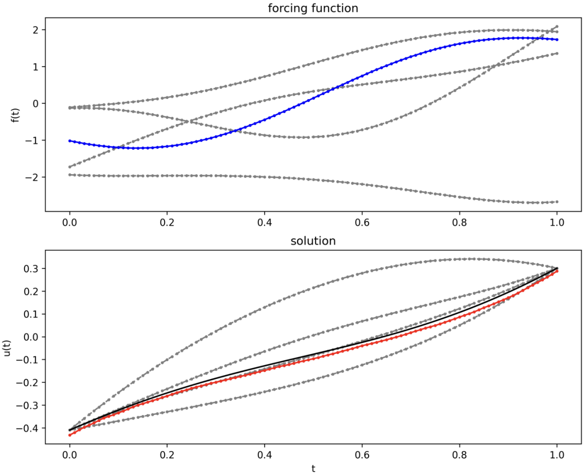
- Inverse problem: Given the solution $u(x)$, find the coefficient function $f(x)$.

- This time, the parameters a and c *and* boundary values u_0 and u_1 are unknown and must be learned in-context.

Experiment Details

- The prompt contains 4 examples and 1 query of input/output functions (during both training and testing).
- To generate the training data, for each of the four problems:
  - generate 100 different values of the parameters randomly;
  - for each set of parameters, solve the PDE and discretize it at 100 locations.
- Training is done by gradient descent, sampling batches from the training data.
- During testing, a prompt is randomly generated from a randomly selected PDE, with a randomly selected setting of parameters.

Sample Result: Forward Poisson BVP

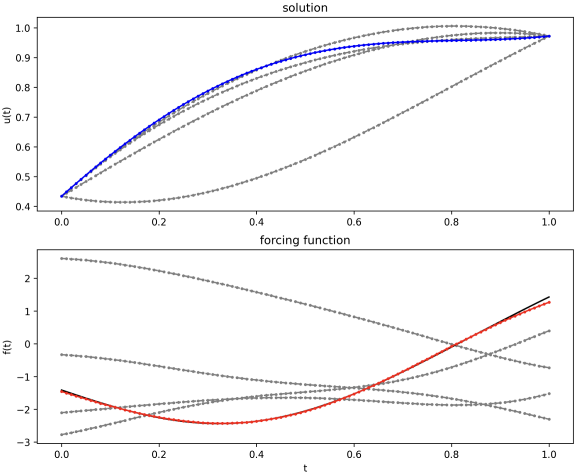


$$\frac{d^2 u(x)}{dx^2} = f(x)$$

$$u(0) = u_0,\ u(1) = u_1$$

examples
query
prediction
groundtruth

Sample Result: Inverse Poisson BVP

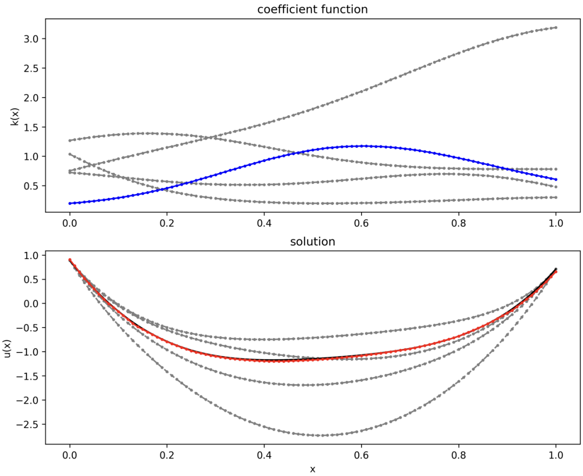


$$\frac{d^2 u(x)}{dx^2} = f(x)$$

$$u(0) = u_0, \; u(1) = u_1$$

examples
query
prediction
groundtruth

Sample Result: Forward Reaction-Diffusion BVP



$$a\frac{d^2u(x)}{dx^2}+k(x)u(x)=c$$

$$u(0)=u_0\,,\ u(1)=u_1$$

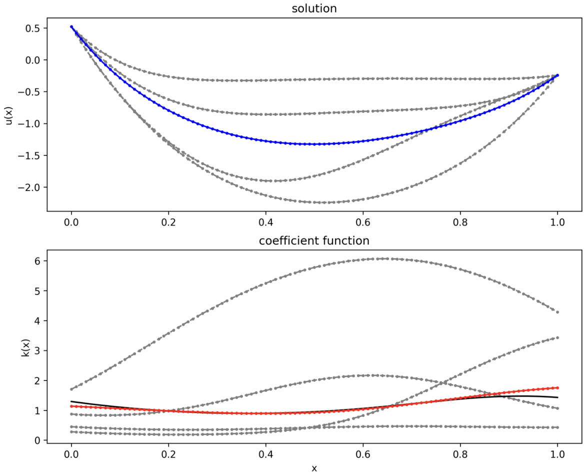
examples

query

prediction

groundtruth

Sample Result: Inverse Reaction-Diffusion BVP



$$a\frac{d^2u(x)}{dx^2} + k(x)u(x) = c$$

$$u(0) = u_0,\; u(1) = u_1$$

examples
query
prediction
groundtruth