DeepONet: Learning nonlinear operators

Lu Lu

joint work with P. Jin, G. Pang, Z. Zhang, & G. Karniadakis

Division of Applied Mathematics, Brown University

SIAM Conference on Mathematics of Data Science

June, 2020
From function to operator

- **Function**: \( \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \)
  - e.g., image classification: \( \sum \to 5 \)

- **Operator**: function (\( \infty \)-dim) \( \mapsto \) function (\( \infty \)-dim)
  - e.g., derivative (local): \( x(t) \mapsto x'(t) \)
  - e.g., integral (global): \( x(t) \mapsto \int K(s, t)x(s)ds \)
  - e.g., dynamic system:
  - e.g., biological system
  - e.g., social system
From function to operator

- **Function:** \( \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2} \)
  - e.g., image classification: \( \sum \rightarrow 5 \)

- **Operator:** function (\( \infty \)-dim) \( \mapsto \) function (\( \infty \)-dim)
  - e.g., derivative (local): \( x(t) \mapsto x'(t) \)
  - e.g., integral (global): \( x(t) \mapsto \int K(s, t)x(s)ds \)
  - e.g., dynamic system:
  - e.g., biological system
  - e.g., social system

⇒ Can we learn operators via neural networks?
⇒ How?
Problem setup

\[ G : u \mapsto G(u) \]
\[ G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R} \]
Universal Approximation Theorem for Operator

\[ G : u \mapsto G(u) \]
\[ G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R} \]

**Theorem (Chen & Chen, 1995)**

Suppose that \( \sigma \) is a continuous non-polynomial function, \( X \) is a Banach Space, \( K_1 \subset X, K_2 \subset \mathbb{R}^d \) are two compact sets in \( X \) and \( \mathbb{R}^d \), respectively, \( V \) is a compact set in \( C(K_1) \), \( G \) is a continuous operator, which maps \( V \) into \( C(K_2) \). Then for any \( \epsilon > 0 \), there are positive integers \( n, p, m, \)

constants \( c_{ki}^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}, w_k \in \mathbb{R}^d, x_j \in K_1, i = 1, \ldots, n, k = 1, \ldots, p, j = 1, \ldots, m, \) such that

\[
\left| G(u)(y) - \sum_{k=1}^{p} \sum_{i=1}^{n} c_{ki}^k \sigma \left( \sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right) \sigma(w_k \cdot y + \zeta_k) \right| < \epsilon
\]

holds for all \( u \in V \) and \( y \in K_2 \).
Convergence w.r.t. the number of sensors

Consider $G : u(x) \mapsto s(x)$ ($x \in [0, 1]$) by ODE system

$$\frac{d}{dx} s(x) = g(s(x), u(x), x), \quad s(0) = s_0$$

$\forall u \in V \Rightarrow u_m \in V_m$

Let $\kappa(m, V) := \sup_{u \in V} \max_{x \in [0, 1]} |u(x) - u_m(x)|$

e.g., Gaussian process with kernel $e^{-\frac{\|x_1 - x_2\|^2}{2l^2}} : \kappa(m, V) \sim \frac{1}{m^2l^2}$

Theorem (Lu et al., 2019; informal)

There exists a constant $C$, such that for any $y \in [0, 1]$,

$$\sup_{u \in V} \|G(u)(y) - NN(u(x_1), \ldots, u(x_m), y)\|_2 < C\kappa(m, V).$$
Problem setup

\[ G : u \mapsto G(u) \]
\[ G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R} \]

- Inputs: \( u \) at sensors \( \{x_1, x_2, \ldots, x_m\} \), \( y \in \mathbb{R}^d \)
- Output: \( G(u)(y) \in \mathbb{R} \)

Next, how to design the network?

---

Lu et al., arXiv:1910.03193, 2019
Deep operator network (DeepONet)

C Stacked DeepONet

\[ u \rightarrow \begin{array}{c}
\text{Branch net}_1 \rightarrow b_1 \\
\text{Branch net}_2 \rightarrow b_2 \\
\vdots \\
\text{Branch net}_p \rightarrow b_p
\end{array} \rightarrow G(u)(y) \rightarrow \begin{array}{c}
\text{Trunk net}_1 \rightarrow t_1 \\
\text{Trunk net}_2 \rightarrow t_2 \\
\vdots \\
\text{Trunk net}_p \rightarrow t_p
\end{array} \]

D Unstacked DeepONet

\[ u \rightarrow \begin{array}{c}
\text{Branch net} \rightarrow \left( \begin{array}{c}
u(x_1) \\
u(x_2) \\
\vdots \\
u(x_m)
\end{array} \right)
\end{array} \rightarrow G(u)(y) \rightarrow \begin{array}{c}
\text{Trunk net}_1 \rightarrow t_1 \\
\text{Trunk net}_2 \rightarrow t_2 \\
\vdots \\
\text{Trunk net}_p \rightarrow t_p
\end{array} \]

\[
G(u)(y) \approx \sum_{k=1}^{p} b_k(u) \cdot t_k(y)
\]

Ideas:

- Prior knowledge: \( u \) and \( y \) are independent
- \( G(u)(y) \): a function of \( y \) conditioning on \( u \)
  - \( t_k(y) \): basis functions of \( y \)
  - \( b_k(u) \): \( u \)-dependent coefficients

Lu et al., arXiv:1910.03193, 2019
Explicit operator: A simple ODE case

\[
\frac{ds(x)}{dx} = u(x), \quad x \in [0, 1], \quad s(0) = 0
\]

\[G : u(x) \mapsto s(y) = s(0) + \int_0^y u(\tau) d\tau\]

Very small generalization error!

More problems: 1D Caputo fractional derivative, 2D Riesz fractional Laplacian

Lu et al., arXiv:1910.03193, 2019
Implicit operator: A nonlinear ODE case

\[ \frac{ds(x)}{dx} = -s^2(x) + u(x) \]

Linear correlation between training and test errors

- **A**: in one training process
- **B**: across multiple runs (random dataset and network initialization)

---

Lu et al., arXiv:1910.03193, 2019
Implicit operator: Gravity pendulum with an external force

\[ \frac{ds_1}{dt} = s_2, \quad \frac{ds_2}{dt} = -k \sin s_1 + u(t) \]

\[ G : u(x) \mapsto s(x) \]

Test/generalization error:
- small dataset: exponential convergence
- large dataset: polynomial rates
- larger network has later transition point

Lu et al., arXiv:1910.03193, 2019
Implicit operator: Diffusion-reaction system

\[ \frac{\partial s}{\partial t} = D \frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), \quad x \in [0, 1], t \in [0, 1] \]

# Training points = \#u \times P
Implicit operator: Diffusion-reaction system

$$\frac{\partial s}{\partial t} = D \frac{\partial^2 s}{\partial x^2} + ks^2 + u(x), \quad x \in [0, 1], t \in [0, 1]$$

# Training points = #u × P

Small dataset: Exponential convergence

Large dataset: Polynomial convergence

Lu et al., arXiv:1910.03193, 2019
Advection-diffusion system

\[ \frac{\partial s}{\partial t} + \frac{\partial s}{\partial x} - \frac{\partial^2 s}{\partial x^2} = 0 \quad \text{or} \quad \frac{\partial s}{\partial t} - 1.5(RL_{\infty}D_x^{1.5})s = 0 \]

\( x \in [0, 1], \ t \in [0, 1], \) periodic BC

\( G : u(x) = s(x, 0) \mapsto s(x, t) \)
- 100 \( u \) sensors
- training data: \# IC = 1000, 100 random points of \( s(x, t) \) for each IC

Prediction for a new random IC:
Advection-diffusion system

More predictions:

A random IC

Error of $s(x, t)$
Consider the population growth model

\[ dy(t; \omega) = k(t; \omega)y(t; \omega)dt, \quad y(0) = 1 \]

Stochastic process \( k(t; \omega) \sim \mathcal{GP}(0, \sigma^2 \exp(-\|t_1 - t_2\|^2 / 2l^2)) \)

**Goal:** Given a new \( k(t; \omega) \), predict the stochastic solution \( y(t; \omega) \)

**Ideas:**
- Karhunen-Loève (KL) expansion: \( k(t; \omega) \approx \sum_{i=1}^{N} \sqrt{\lambda_i} e_i(t) \xi_i(\omega) \)
- branch net inputs: \([\sqrt{\lambda_1} e_1(t), \sqrt{\lambda_2} e_2(t), \ldots, \sqrt{\lambda_N} e_N(t)] \in \mathbb{R}^{N \times m}\)
  where \( \sqrt{\lambda_i} e_i(t) = \sqrt{\lambda_i} [(e_i(t_1), e_i(t_2)), \ldots, e_i(t_m)] \in \mathbb{R}^{m}\)
- trunk net inputs: \([t, \xi_1, \xi_2, \ldots, \xi_N] \in \mathbb{R}^{N+1}\)
Stochastic ODE

- Choose $N = 5$ to conserve 99.9% stochastic energy
- Train with 10000 different $k(t; \omega)$ with $l$ randomly sampled in $[1, 2]$, and for each $k(t; \omega)$ we use only one realization
- Test MSE is $8.0 \times 10^{-5} \pm 3.4 \times 10^{-5}$

Example: 10 different random samples from $k(t; \omega)$ with $l = 1.5$
Consider the elliptic problem with multiplicative noise

\[-\text{div}(e^{b(x;\omega)}\nabla u(x;\omega)) = f(x), \quad x \in (0, 1)\]

with zero Dirichlet boundary conditions, \( f(x) = 10 \).

Stochastic process \( b(x; \omega) \sim \mathcal{GP}(0, \sigma^2 \exp(-\|x_1 - x_2\|^2/2l^2)) \)
Diverse applications:

- Explicit operators: integral operators, fractional derivative, fractional Laplacian
- Implicit operators: (linear or nonlinear) ODE/PDE system, stochastic ODE/PDE

Good performance & data efficiency:

- Small generalization error
- Exponential/polynomial error convergence