DeepONet: Learning nonlinear operators

Lu Lu

Applied Mathematics Instructor
Department of Mathematics, MIT

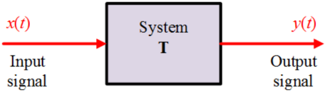
SIAM Conference on Computational Science and Engineering
Mar 3, 2021

From function to operator

- Function: $\mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$

  e.g., image classification:  $\mapsto$ 5

- Operator: function (∞-dim) $\mapsto$ function (∞-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: 

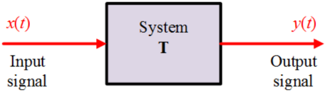
  e.g., biological system
  e.g., social system

From function to operator

- Function: $\mathbb{R}^{d_1} \to \mathbb{R}^{d_2}$

  e.g., image classification:  $\mapsto$ 5

- Operator: function (∞-dim) $\mapsto$ function (∞-dim)
  e.g., derivative (local): $x(t) \mapsto x'(t)$
  e.g., integral (global): $x(t) \mapsto \int K(s,t)x(s)ds$

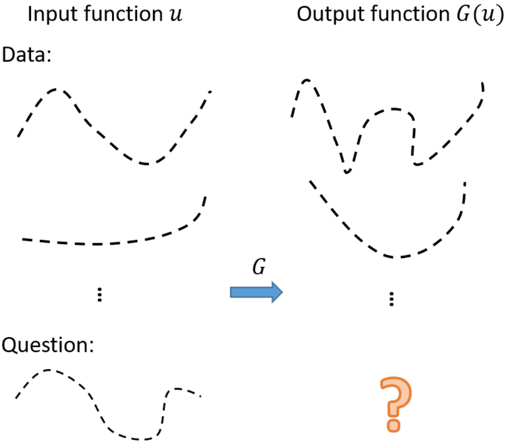
  e.g., dynamic system: 

  e.g., biological system
  e.g., social system

  $\Rightarrow$ Can we learn operators via neural networks?
  $\Rightarrow$ How?

Problem setup

$$G : u \in V \subset C(K_1) \mapsto G(u) \in C(K_2)$$
$$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$$



Universal Approximation Theorem for Operator

$G : u \in V \subset C(K_1) \mapsto G(u) \in C(K_2)$
$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$

Theorem (Chen & Chen, 1995)

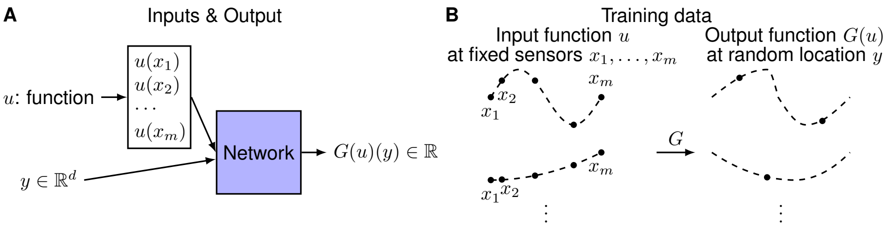
Suppose that σ is a continuous non-polynomial function, X is a Banach Space, $K_1 \subset X$, $K_2 \subset \mathbb{R}^d$ are two compact sets in X and $\mathbb{R}^d$, respectively, V is a compact set in $C(K_1)$, G is a continuous operator, which maps V into $C(K_2)$.
Then for any $\epsilon > 0$, there are positive integers n, p, m, constants $c_i^k, \xi_{ij}^k, \theta_i^k, \zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \ldots, n$, $k = 1, \ldots, p$, $j = 1, \ldots, m$, such that

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

holds for all $u \in V$ and $y \in K_2$.

Problem setup

$$G : u \in V \subset C(K_1) \mapsto G(u) \in C(K_2)$$
$$G(u) : y \in \mathbb{R}^d \mapsto G(u)(y) \in \mathbb{R}$$

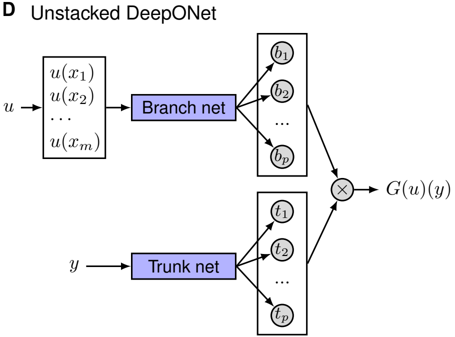


- Inputs: u at sensors $\{x_1, x_2, \ldots, x_m\} \in \mathbb{R}^m$, $y \in \mathbb{R}^d$
- Output: $G(u)(y) \in \mathbb{R}$

Lu et al., *Nature Mach Intell*, 2021

Deep operator network (DeepONet)

$$G(u)(y) \approx \sum_{k=1}^{p} \underbrace{b_k(u)}_{\text{branch}} \underbrace{t_k(y)}_{\text{trunk}}$$



Idea: $G(u)(y)$: a function of y conditioning on u

- $t_k(y)$: basis function of y
- $b_k(u)$: u-dependent coefficient, i.e., functional of u

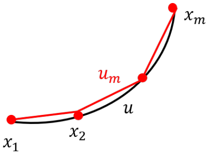
Lu et al., *Nature Mach Intell*, 2021

Error analysis: the number of sensors

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

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

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



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

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

Theorem (Lu et al., *Nature Mach Intell*, 2021)

Assume $\mathbf{g}$ *is Lipschitz continuous* (c)*.*
Then there exists a DeepONet $\mathcal{N}$*, such that*

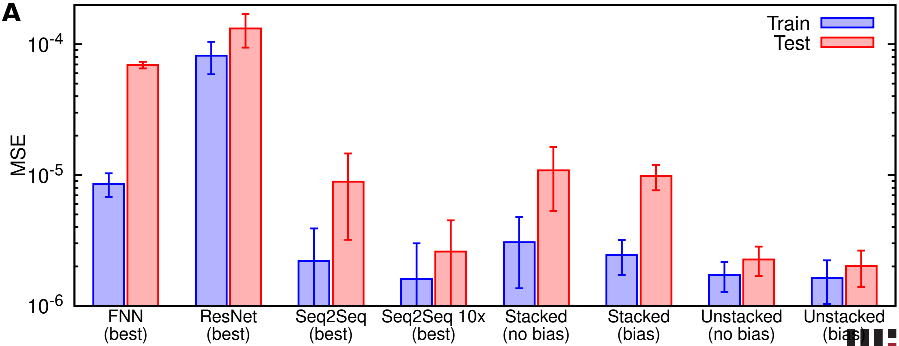
$$\sup_{u\in V} \max_{x\in[0,1]} \|G(u)(x) - \mathcal{N}([u(x_1), \ldots, u(x_m)], x)\|_2 < ce^c \kappa(m, V).$$

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!

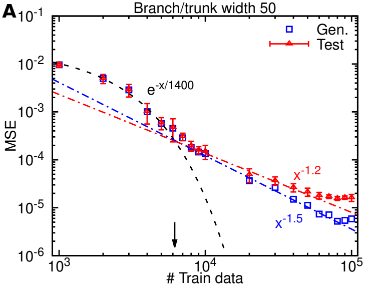


Lu et al., *Nature Mach Intell*, 2021

Implicit operator: Gravity pendulum with an external force

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

$G : u(t) \mapsto \mathbf{s}(t)$



Test/generalization error:

- small dataset: exponential convergence
- large dataset: polynomial rates
- larger network has later transition point

Lu et al., *Nature Mach Intell*, 2021

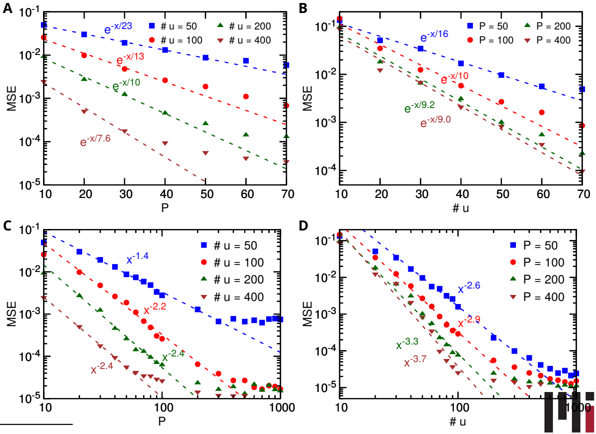
Implicit operator: Diffusion-reaction system

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

$\#$ Training points $= \#u \times P$

Small dataset:
Exponential convergence

Large dataset:
Polynomial convergence



Lu et al., *Nature Mach Intell*, 2021

Stochastic PDE

Consider the elliptic problem with multiplicative noise

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

with zero boundary conditions, $f(x) = 10$.
Stochastic process $b(x;\omega) \sim \mathcal{GP}(0, \sigma^2 \exp(-\|x_1 - x_2\|^2/2l^2))$

$$G : b(x;\omega) \mapsto u(x;\omega)$$

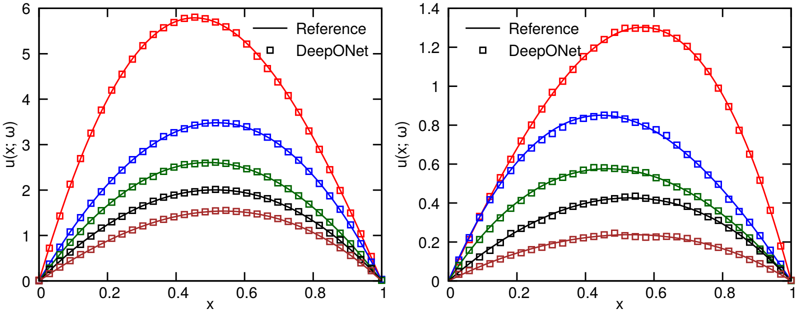
Ideas:

- Karhunen-Loève (KL) expansion: $b(x;\omega) \approx \sum_{i=1}^{N} \sqrt{\lambda_i} e_i(x) \xi_i(\omega)$
- branch net inputs: $[\sqrt{\lambda_1} e_1(x), \sqrt{\lambda_2} e_2(x), \ldots, \sqrt{\lambda_N} e_N(x)] \in \mathbb{R}^{N \times m}$, where $\sqrt{\lambda_i} e_i(x) = \sqrt{\lambda_i}[e_i(x_1), e_i(x_2), \ldots, e_i(x_m)] \in \mathbb{R}^m$
- trunk net inputs: $[x, \xi_1, \xi_2, \ldots, \xi_N] \in \mathbb{R}^{N+1}$

Lu et al., *Nature Mach Intell*, 2021

Stochastic PDE

Example: 10 different random samples of $u(x;\omega)$ for $b(x;\omega)$ with $l = 1.5$



Lu et al., *Nature Mach Intell*, 2021

DeepONet for learning operators

DeepONet (**Lu** et al., *Nature Mach Intell*, 2021)

- **Algorithms & Applications**
  - 16 ODEs/PDEs (nonlinear, fractional & stochastic)
  - Multiphysics & Multiscale problems
    - Bubble growth dynamics from nm to mm (*J Chem Phys*, 2021)
    - Hypersonics (arXiv:2011.03349)
    - Electroconvection (arXiv:2009.12935)
- **Observation**: Good performance
  - Small generalization error
  - Exponential/polynomial error convergence
- **Theory**
  - Convergence rate for advection-diffusion equations (arXiv:2102.10621)

DeepM&Mnet: Multiphysics & Multiscale problems

Hypersonics: Coupled flow and finite-rate chemistry behind a normal shock (arXiv:2011.03349)