NEURAL OPERATORS FOR SCIENTIFIC MACHINE LEARNING

Presented by Handi Zhang

Functions map data, Operators map functions

- 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): $x(t) \mapsto \int K(s,t)x(s)ds$

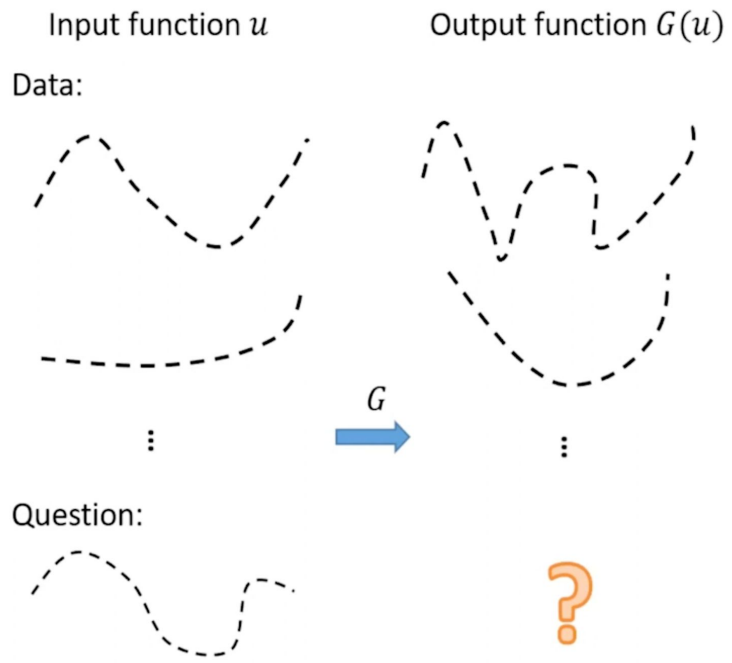
  e.g., dynamic system: 

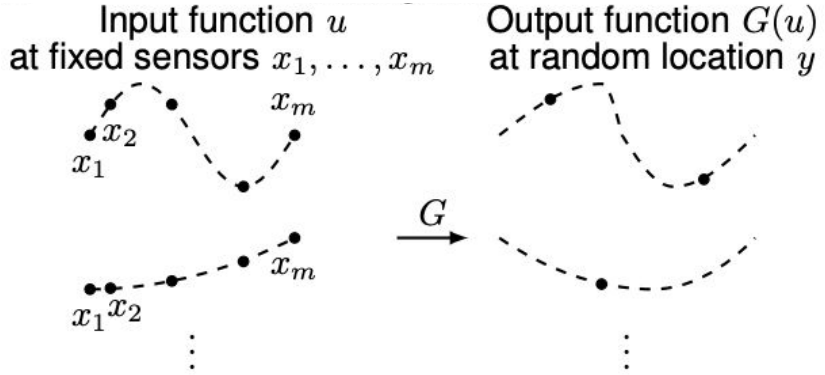
  e.g., biological system
  e.g., social system

Problem Setup

$G : u \mapsto G(u)$

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





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

- Inputs: u at sensors $\{x_1, x_2, \ldots, x_m\}, y$
- Outputs: $G(u)(y)$
- $G(u)(y)$: a function of y conditioning on u
  - $t_k(y)$: basis functions of y
  - $b_k(u)$: u-dependent coefficient

Universal Approximation Theorem for Operator

Theorem 1 (Universal Approximation Theorem for Operator). *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 nonlinear continuous operator, which maps V into $C(K_2)$. Then for any $\epsilon > 0$, there are positive integers n, p and m, constants c_i^k, ξ_{ij}^k, θ_i^k, $\zeta_k \in \mathbb{R}$, $w_k \in \mathbb{R}^d$, $x_j \in K_1$, $i = 1, \ldots, n$, $k = 1, \ldots, p$ and $j = 1, \ldots, m$, such that*

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

holds for all $u \in V$ and $y \in K_2$. Here, $C(K)$ is the Banach space of all continuous functions defined on K with norm $\| f \|_{C(K)} = \max_{x \in K} |f(x)|$.

Generalized Universal Approximation Theorem for Operator

Theorem 2 (Generalized Universal Approximation Theorem for Operator). *Suppose that 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 $C(K_1)$. Assume that $G: V \to C(K_2)$ is a nonlinear continuous operator. Then, for any $\epsilon > 0$, there exist positive integers m, p, continuous vector functions $\mathbf{g}: \mathbb{R}^m \to \mathbb{R}^p$, $\mathbf{f}: \mathbb{R}^d \to \mathbb{R}^p$, and $x_1, x_2, \ldots, x_m \in K_1$, such that*

$$\left| G(u)(y) - \langle \underbrace{\mathbf{g}(u(x_1), u(x_2), \cdots, u(x_m))}_{\text{branch}}, \underbrace{\mathbf{f}(y)}_{\text{trunk}} \rangle \right| < \epsilon$$

holds for all $u \in V$ and $y \in K_2$, where $\langle \cdot, \cdot \rangle$ denotes the dot product in $\mathbb{R}^p$. Furthermore, the functions $\mathbf{g}$ and $\mathbf{f}$ can be chosen as diverse classes of neural networks, which satisfy the classical universal approximation theorem of functions, for example, (stacked/unstacked) fully connected neural networks, residual neural networks and convolutional neural networks.

Sketch of Proof

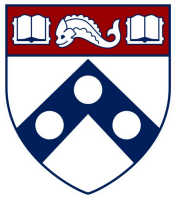
Continuously extend $\mathcal{G}(u)(y), y \in K_2$ to $\mathcal{G}(u)(y), y \in D$.

$$
\begin{aligned}
\mathcal{G}(u) &\approx \mathcal{G}(\mathcal{I}^0_{m,x}u) \quad \text{(piecewise constant interpolation)} \\
&\approx \sum_{k=1}^{p} \int_D \mathcal{G}(\mathcal{I}^0_{m,x}u) e_k(y) dy \; e_k(y), \quad (K_2 \subset D, \text{spectral expansion}) \\
&\approx \sum_{k=1}^{p} \int_D \mathcal{I}^0_{n,y}(\mathcal{G}(\mathcal{I}^0_{m,x}u)(y)) e_k(y) dy e_k(y) \quad \text{(interpolation)} \\
&= \sum_{k=1}^{p} (\sum_{i=1}^{n} \mathcal{G}(\mathcal{I}^0_{m,x}u)(y_i) \underbrace{\int_D e_k(y)\chi_{D_i}(y)dy}_{c_i^k}) e_k(y)
\end{aligned}
$$

- $\mathcal{G}\left(\mathcal{I}^0_m u\right)(y_i)$ is continuous in u_m, on $[-M, M]^m$, $M = \max_{1 \le i \le m} |u(x_i)|$.
- As $\mathcal{G} : V \to C(D)$ is continuous, we have uniform approximation

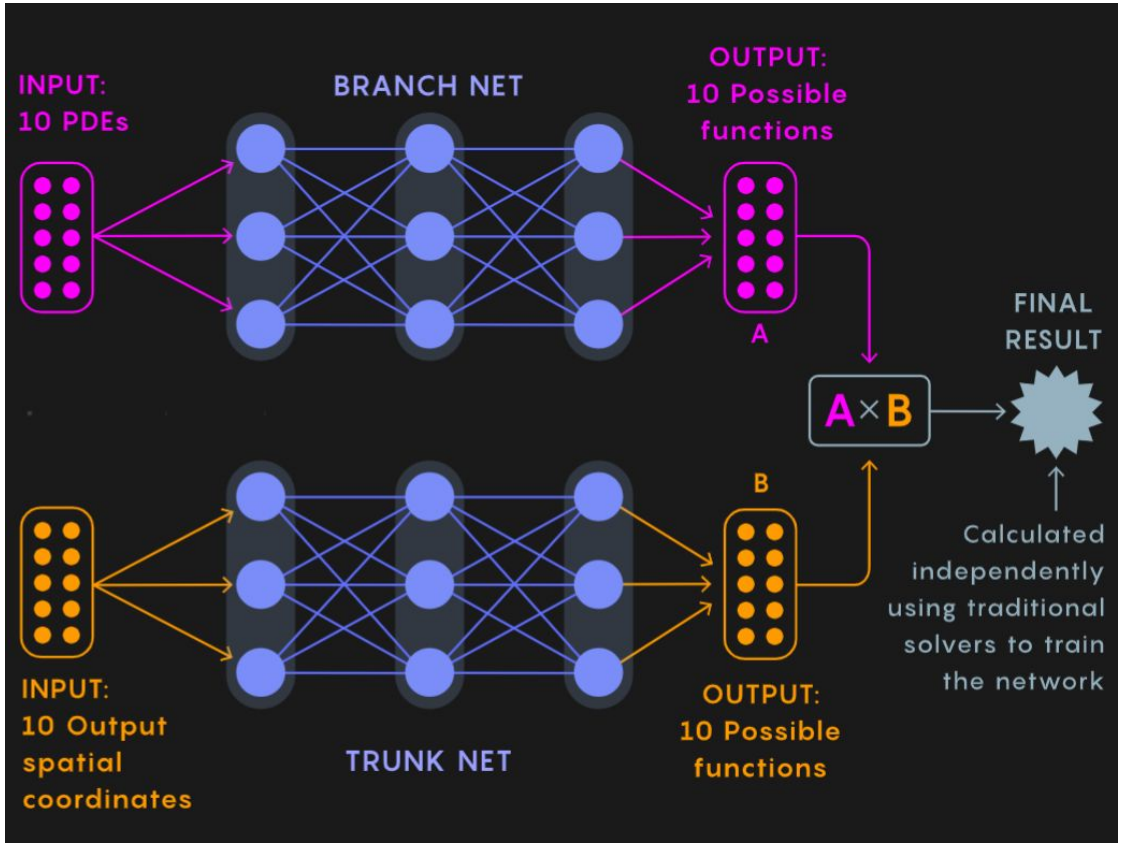
$$
\sup_{u \in V} \sup_{u_m \in [-M,M]^m} \left| \mathcal{G}\left(\mathcal{I}^0_m u\right)(y_i) - g^{\mathcal{N}}\left(u_m; \Theta^{(k,i)}\right) \right| < \epsilon
$$

- $e_k(y) \approx f^{\mathcal{N}}\left(y; \theta^{(k)}\right)$

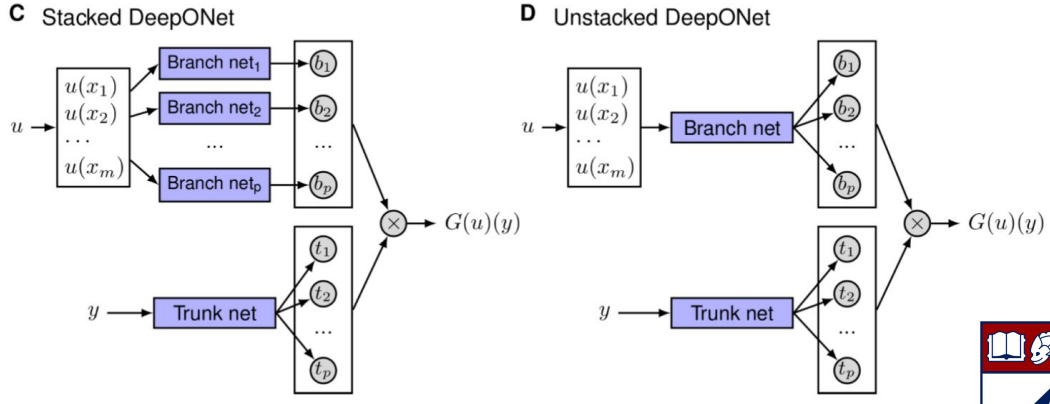


B. Deng, Y. Shin, L. Lu, Z. Zhang, & G. E. Karniadakis. Convergence rate of DeepONets for learning operators arising from advection-diffusion equations. arXiv preprint arXiv:2102.10621, 2021.

DeepONet



$$G(u)(y) \approx \underbrace{\sum_{k=1}^{p} \sum_{i=1}^{n} c_i^k \sigma \left(\sum_{j=1}^{m} \xi_{ij}^k u(x_j) + \theta_i^k \right)}_{branch} \underbrace{\sigma(w_k \cdot y + \zeta_k)}_{trunk}$$



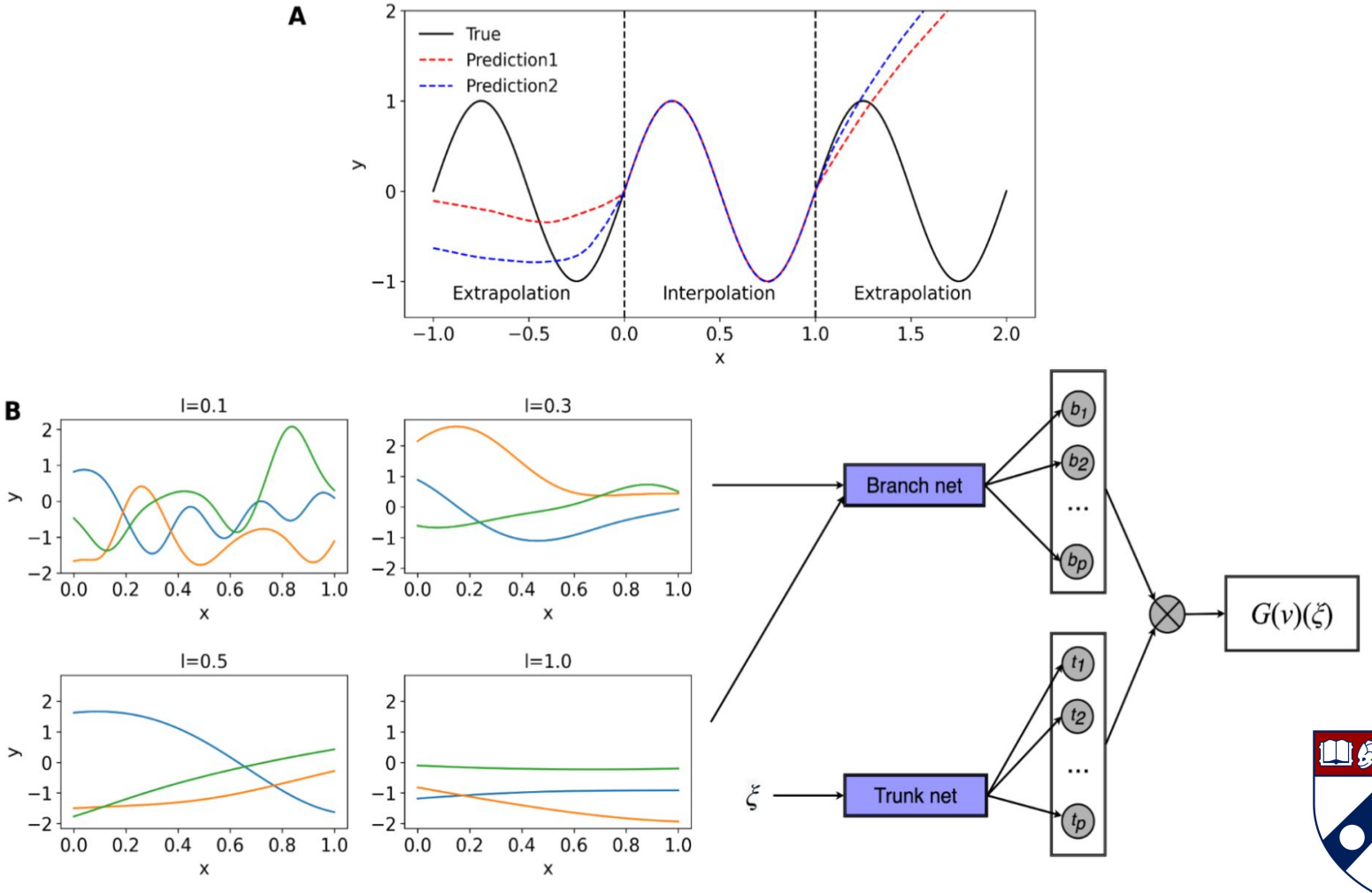
Reliable learning of DeepONets informed by physics or Sparse observation for safe extrapolation

DL models are generally limited to **interpolation**

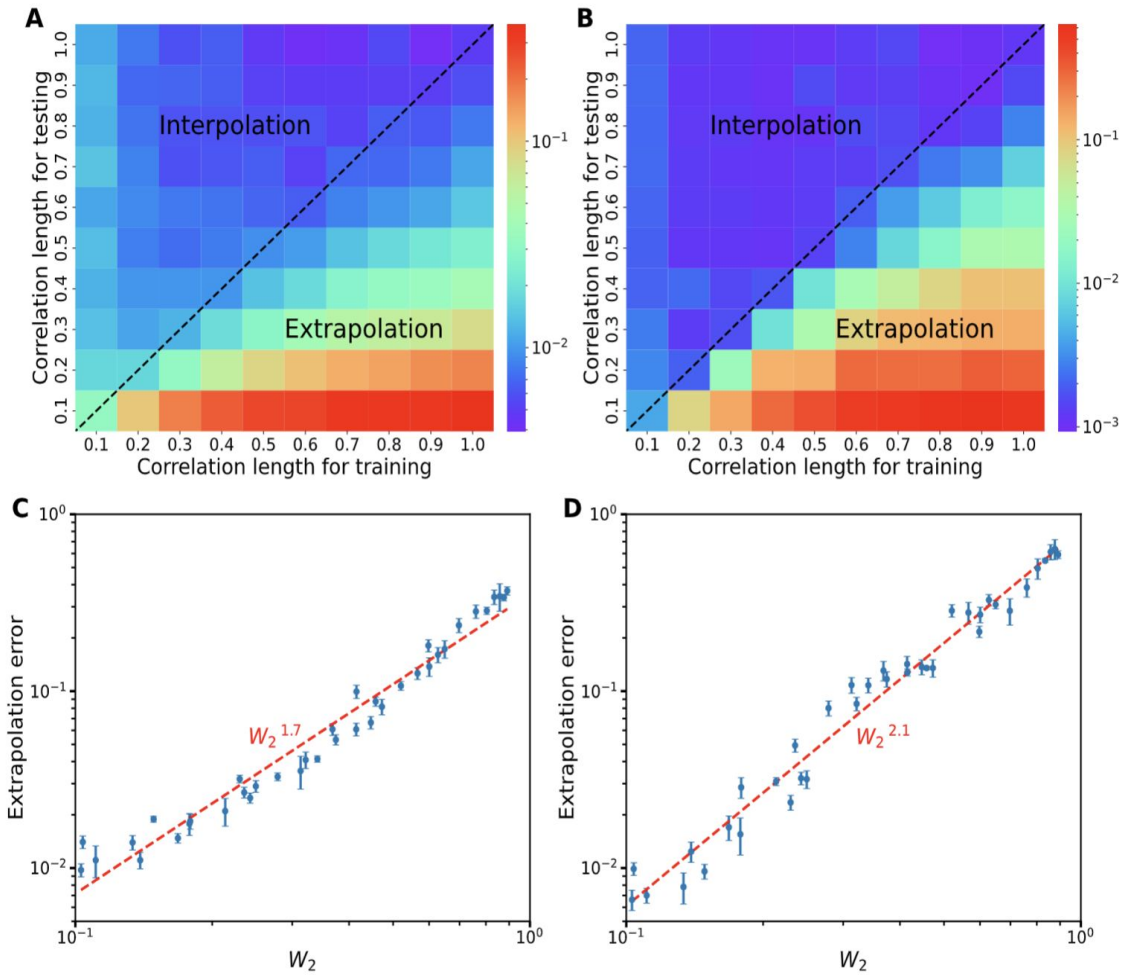
Inference stage of real-world application: **Extrapolations**

This study proposes:

- systematic study of extrapolation for DeepONet
- methods to improve accuracy



Extrapolation of DeepONet



- **Interpolation and extrapolation region**

$$\text{Prediction} = \begin{cases} \text{Interpolation}, & \text{when } l_{\text{train}} \leq l_{\text{test}}, \\ \text{Extrapolation}, & \text{when } l_{\text{train}} > l_{\text{test}}. \end{cases}$$

- **Extrapolation complexity by 2-Wasserstein distance**

Suppose two Gaussian processes,

$$f_1 \sim \mathcal{GP}(m_1, k_1) \text{ and } f_2 \sim \mathcal{GP}(m_2, k_2)$$

A covariance operator is defined as

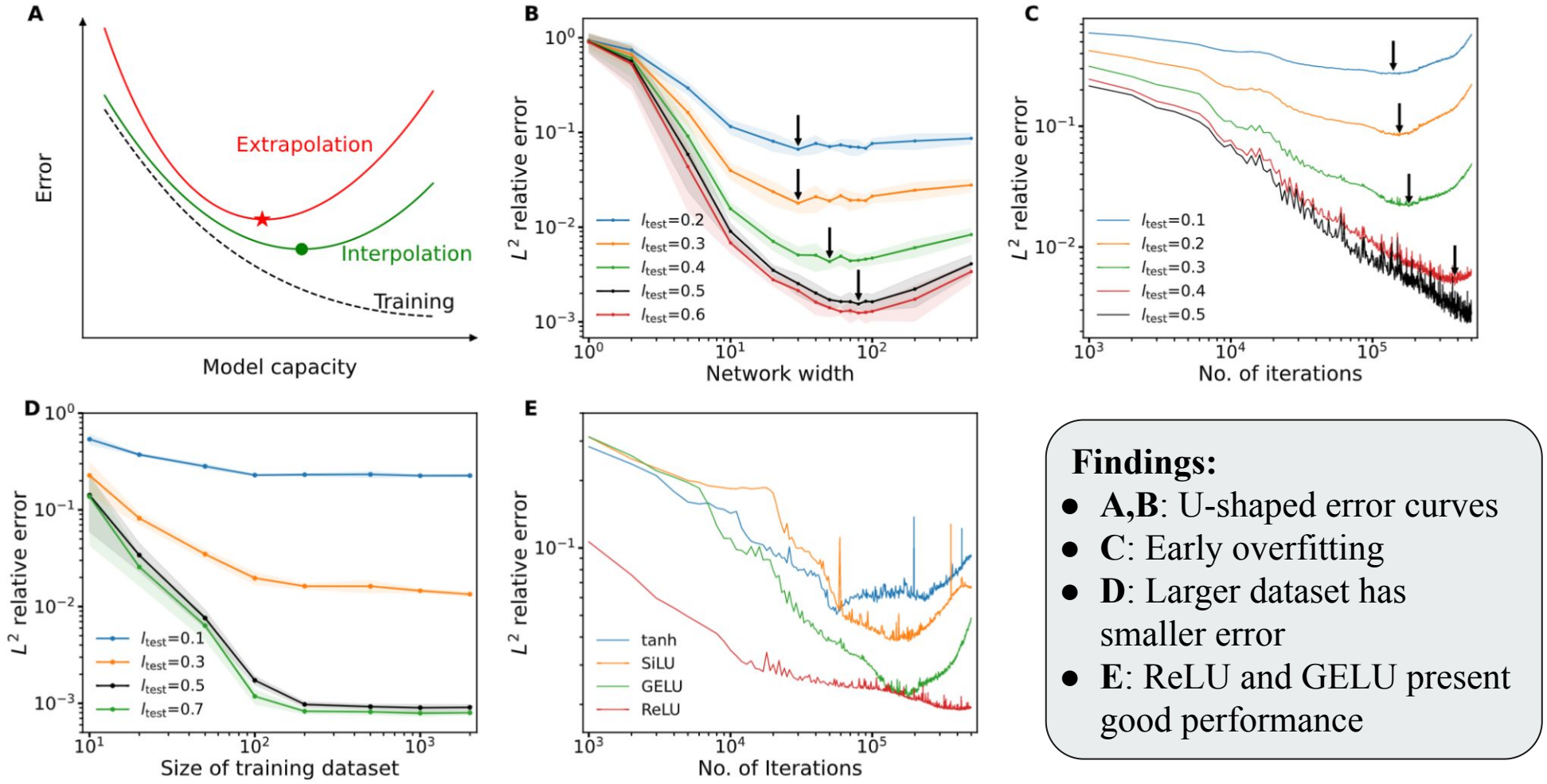
$$[K_i\phi](x) = \int_X k_i(x,s)\phi(s)ds, \quad \forall \phi \in L^2(X).$$

The metric is defined as

$$W_2(f_1, f_2) \coloneqq \left\{ \|m_1 - m_2\|_2^2 + \text{Tr}\left[K_1 + K_2 - 2\left(K_1^{\frac{1}{2}} K_2 K_1^{\frac{1}{2}} \right)^{\frac{1}{2}} \right] \right\}^{\frac{1}{2}}$$

- **Understanding extrapolation error**
  - Bias-variance trade-off
  - Training dataset size
  - Activation functions

Understanding Extrapolation Error



Findings:

- **A,B**: U-shaped error curves
- **C**: Early overfitting
- **D**: Larger dataset has smaller error
- **E**: ReLU and GELU present good performance

Learning methods for safe extrapolation

- **Workflow of extrapolation**
  - **Physics:**

$$\mathcal{F}[u;v]=0 \quad \mathcal{B}[u;v]=0.$$

⇨ Fine-tune with physics

  - **Sparse observations**

$$\mathcal{D}=\{(\xi_1,u(\xi_1)),(\xi_2,u(\xi_2)),\ldots,(\xi_{N_{\text{obs}}},u(\xi_{N_{\text{obs}}}))\}$$

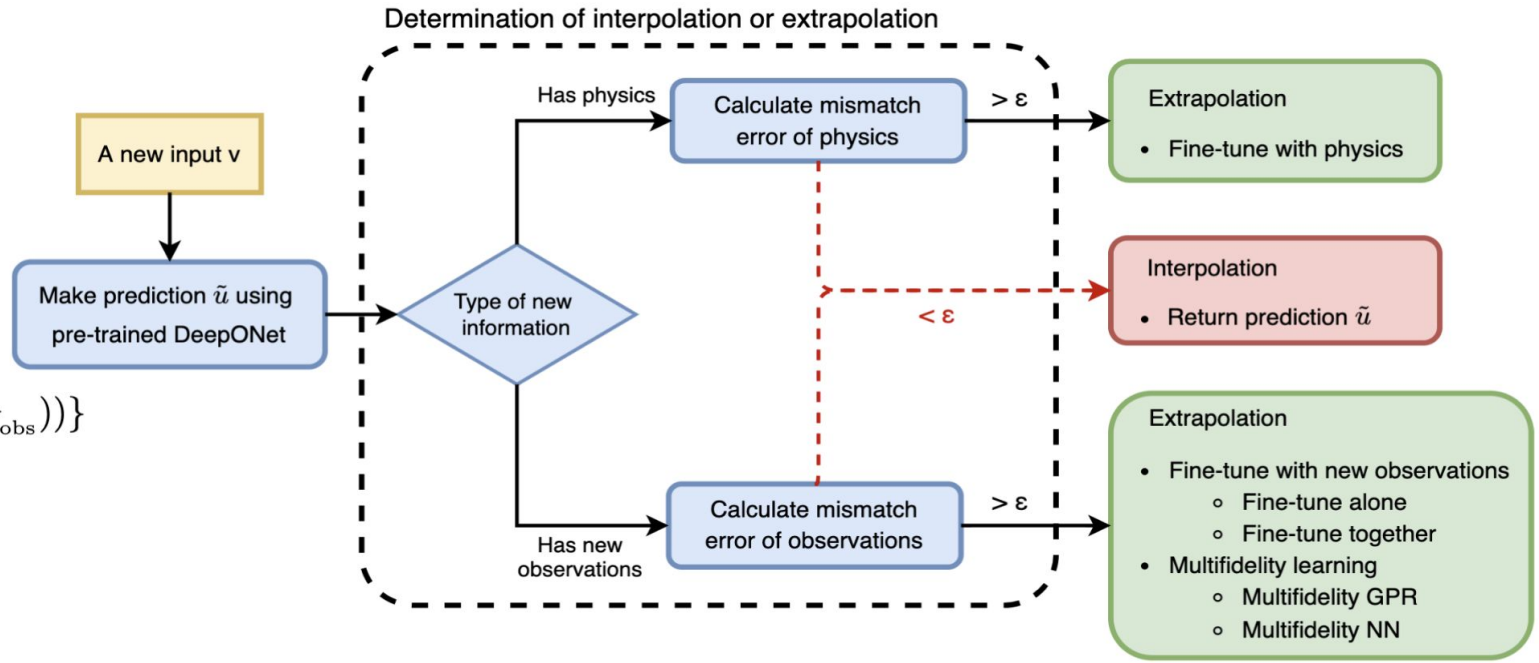
⇨ Fine-tune with new observations

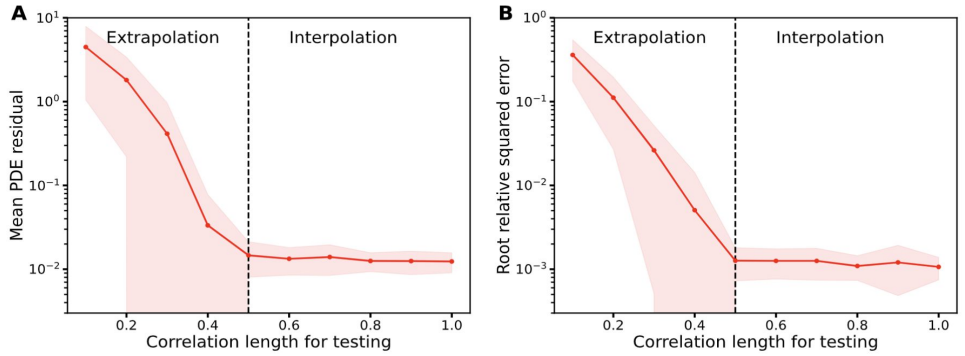
⇨ Multifidelity learning

  - **Errors of mismatch**

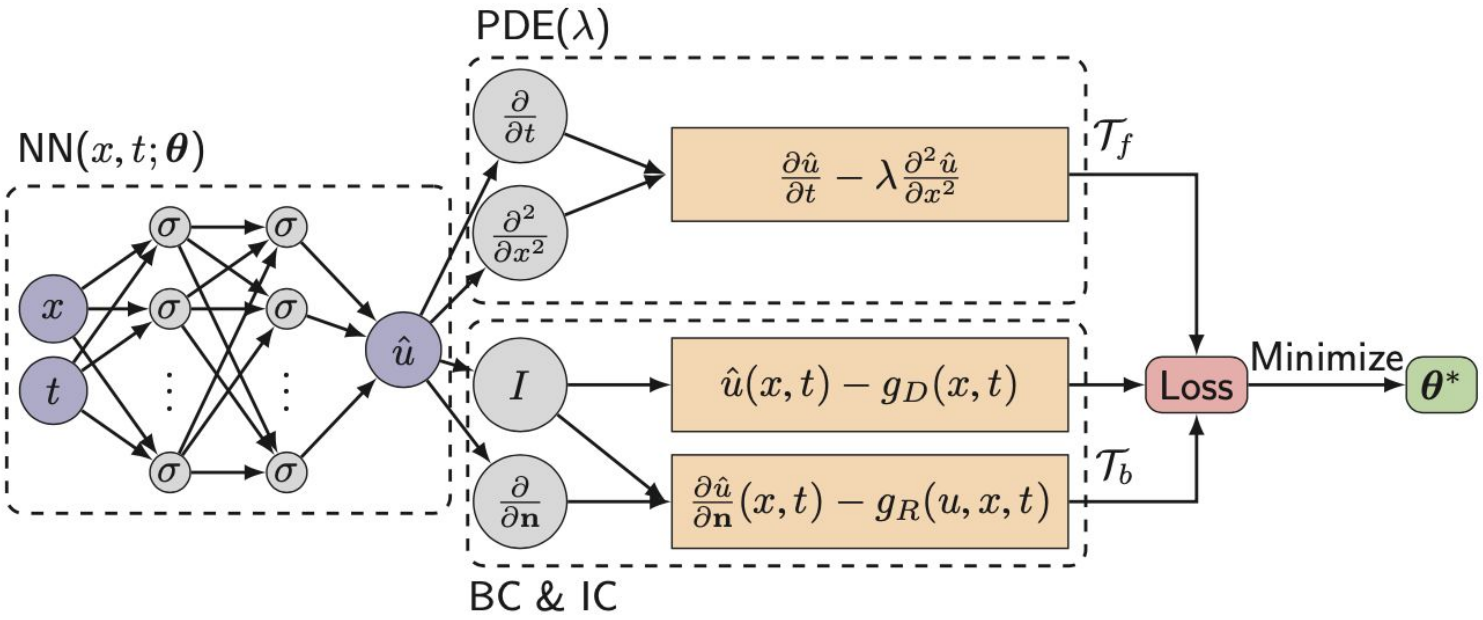
$$\mathcal{E}_{\text{phys}}=\frac{1}{\text{Area}(\Omega)}\int_{\Omega}|\mathcal{F}[\tilde{u};v]|d\xi,$$

$$\mathcal{E}_{\text{obs}}=\sqrt{\frac{\sum_{i=1}^{N_{\text{obs}}}(\tilde{u}(\xi_i)-u(\xi_i))^2}{\sum_{i=1}^{N_{\text{obs}}}u^2(\xi_i)}}$$





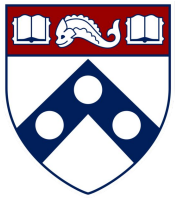
Physics-Informed Neural Networks



$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{T}) = w_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b),$$

$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{T}_f) = \frac{1}{|\mathcal{T}_f|} \sum_{\mathbf{x} \in \mathcal{T}_f} \left\| f\left(\mathbf{x}; \frac{\partial \hat{u}}{\partial x_1}, \ldots, \frac{\partial \hat{u}}{\partial x_d}; \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 \hat{u}}{\partial x_1 \partial x_d}; \ldots; \boldsymbol{\lambda}\right) \right\|_2^2,$$

$$\mathcal{L}_b(\boldsymbol{\theta}; \mathcal{T}_b) = \frac{1}{|\mathcal{T}_b|} \sum_{\mathbf{x} \in \mathcal{T}_b} \|\mathcal{B}(\hat{u}, \mathbf{x})\|_2^2,$$



Reliable learning of DeepONets informed by physics or Sparse observation for safe extrapolation

- **Fine-tuning with Physics (FT-Phys)**

$$\mathcal{L}_{\text{phys}} = w_{\mathcal{F}}\mathcal{L}_{\mathcal{F}} + w_{\mathcal{B}}\mathcal{L}_{\mathcal{B}},$$

- **Fine-tuning with sparse new observations**
  - Fine-tune with new observations alone (FT-Obs-A)

$$\mathcal{L}_{\text{obs}} = \frac{1}{N_{\text{obs}}}\sum_{i=1}^{N_{\text{obs}}}(\mathcal{G}_{\boldsymbol{\theta}}(v)(\xi_i) - u(\xi_i))^2.$$

  - Fine-tune with new observations together (FT-Obs-T)

$$\mathcal{L}_{\mathcal{T},\text{obs}} = \mathcal{L}_{\mathcal{T}} + k\mathcal{L}_{\text{obs}}$$

- **Multifidelity learning**
  - Multifidelity GPR
  - Multifidelity NN

Algorithm 1: Extrapolation via fine-tuning with physics.

Input: A pre-trained DeepONet $\mathcal{G}_{\boldsymbol{\theta}}$, and a new input function v
Data: Physics information $\mathcal{F}$ and $\mathcal{B}$
Output: Prediction $u = \mathcal{G}_{\boldsymbol{\theta}}(v)$

1 Fix the input of the branch net of $\mathcal{G}_{\boldsymbol{\theta}}$ to be v;
2 Sample the two sets of training points $\mathcal{T}_{\mathcal{F}}$ and $\mathcal{T}_{\mathcal{B}}$ for the equation and initial/boundary conditions;
3 Select the weights $w_{\mathcal{F}}$ and $w_{\mathcal{B}}$;
4 Train all the parameters or a subset of the parameters of $\mathcal{G}_{\boldsymbol{\theta}}$ by minimizing the loss $\mathcal{L}_{\text{phys}}$ of Eq. (3);

Algorithm 2: Extrapolation via fine-tuning with new observations.

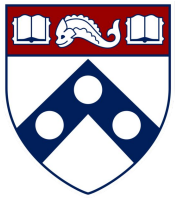
Input: A pre-trained DeepONet $\mathcal{G}_{\boldsymbol{\theta}}$, and a new input function v
Data: New observations $\mathcal{D}$
Output: Prediction $u = \mathcal{G}_{\boldsymbol{\theta}}(v)$

1 Fix the input of the branch net of $\mathcal{G}_{\boldsymbol{\theta}}$ to be v;
2 Fine-tune $\mathcal{G}_{\boldsymbol{\theta}}$ by minimizing the loss of Eq. (4) for fine-tuning alone or Eq. (5) for fine-tuning together;

Algorithm 3: Extrapolation via multifidelity learning with new observations.

Input: A pre-trained DeepONet $\mathcal{G}_{\boldsymbol{\theta}}$, and a new input function v
Data: New observations $\mathcal{D}$
Output: Prediction u

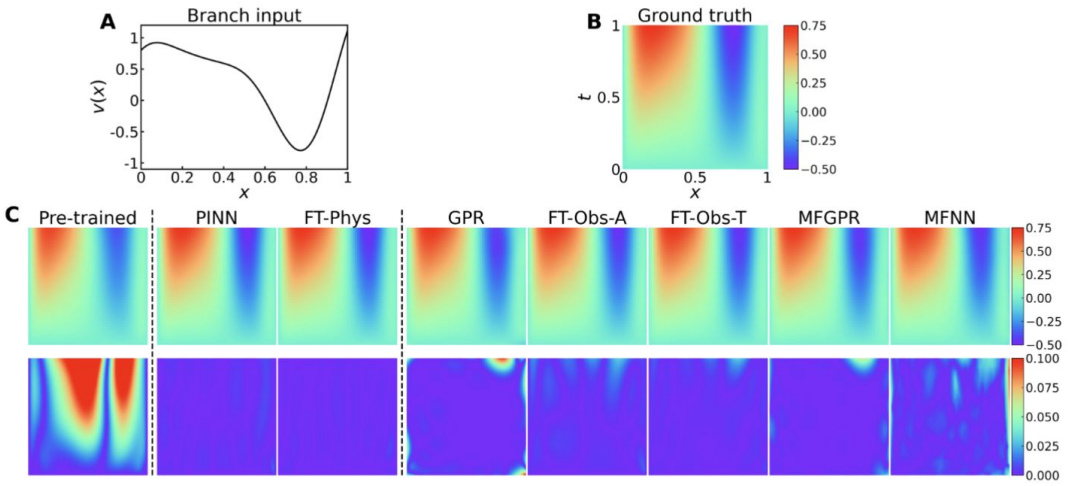
1 Compute the prediction $\tilde{u} = \mathcal{G}_{\boldsymbol{\theta}}(v)$;
2 Sample a set of dense points $\mathcal{S} = \{\xi_i\}_{i=1}^{|\mathcal{S}|}$ and use $\{(\xi_i, \tilde{u}(\xi_i))\}_{i=1}^{|\mathcal{S}|}$ as the low-fidelity dataset;
3 Use $\mathcal{D}$ as the high-fidelity dataset;
4 Train a multifidelity model (MFGPR or MFNN) on the multifidelity datasets;

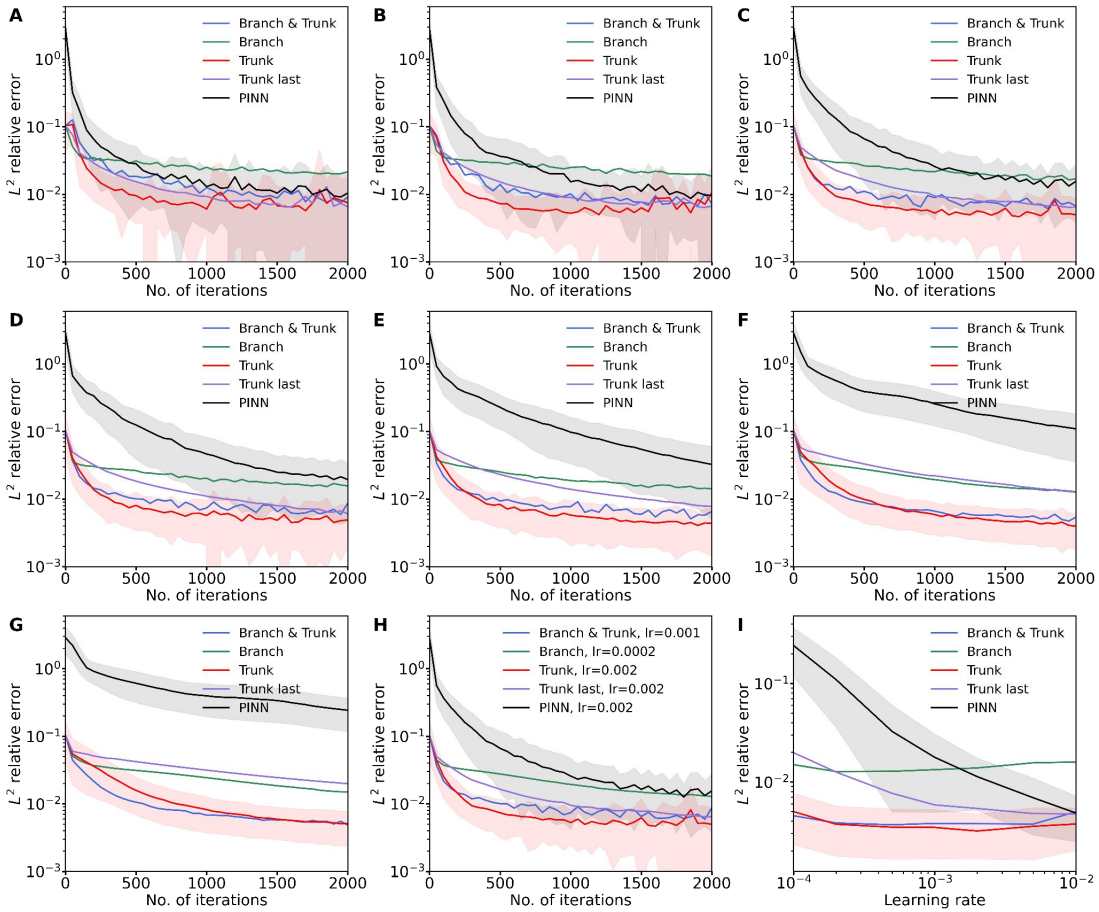


Numerical Results: Diffusion-reaction equation

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

- $D = 0.01$, $k = 0.01$, with zero boundary and initial conditions. $l_{\text{train}} = 0.5$, $l_{\text{test}} = 0.2$
- Mapping from source term $v(x)$ to solution $u(x, t)$.
- $v(x)$ are sampled from a Gaussian random field (GRF) with a radial-basis function kernel (RBF) kernel.





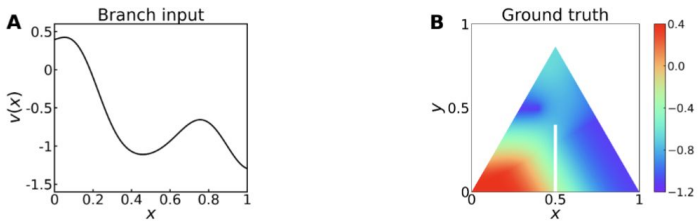
Numerical Results: Other equation

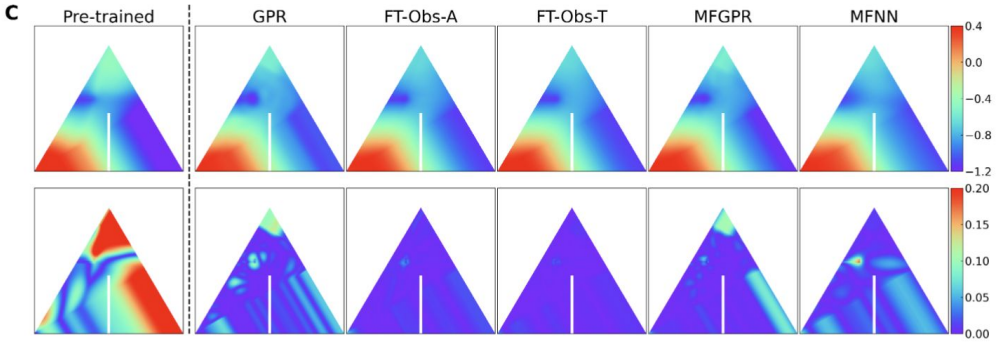
Poisson equation

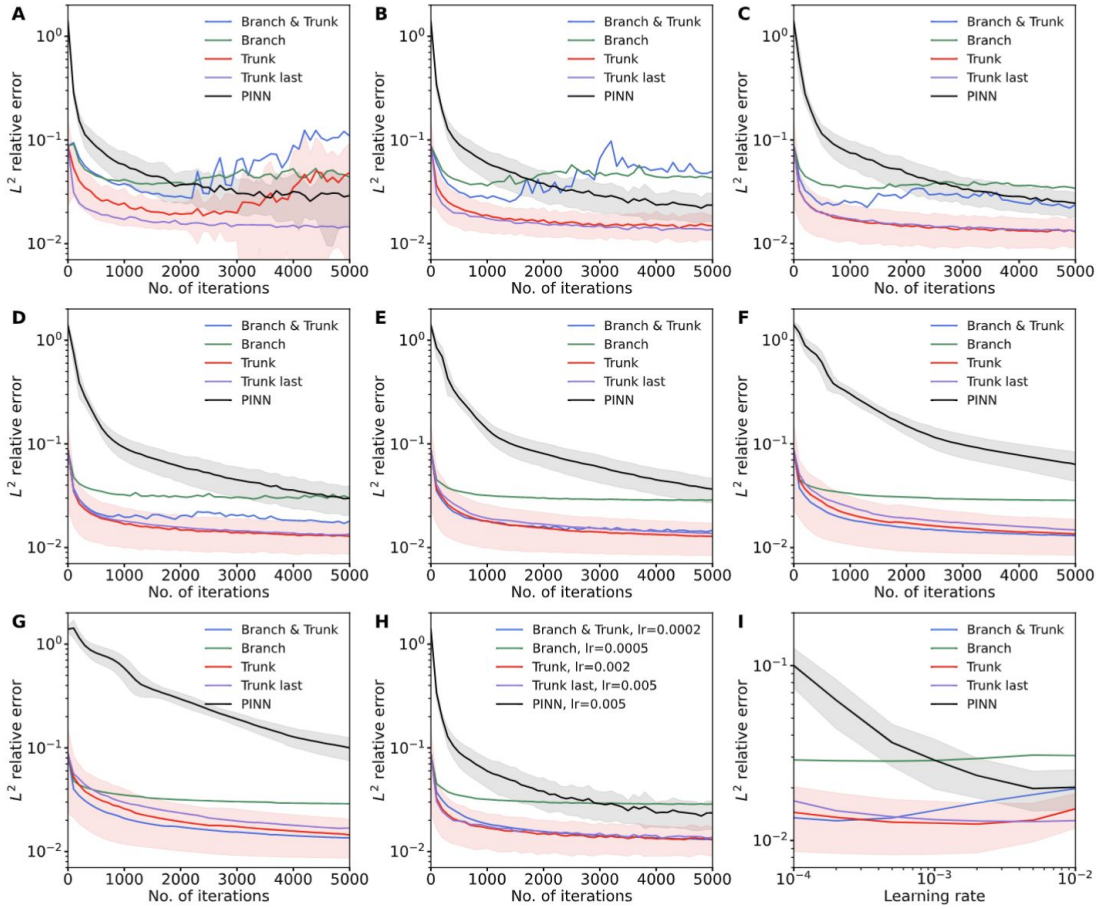
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + 10 = 0, \quad (x, y) \in \Omega,$$

Advection equation

$$\frac{\partial u}{\partial t} + v(x)\frac{\partial u}{\partial x} = 0, \quad x \in [0, 1], \quad t \in [0, 1],$$







Applications of Neural Operators

DeepONet for Approximating Functionals: Predicting unsteady pressure and lift/drag-force coefficients

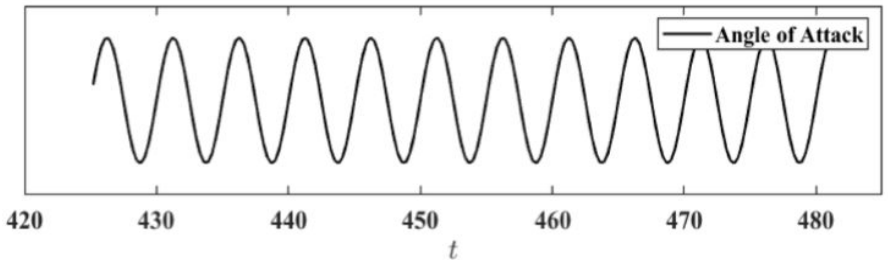
- Simulation of NACA0012 airfoil (Nektar by Z. Wang)

inflow

$$u = U_\infty \cos\left(\frac{\alpha_0 \pi}{180} \times \frac{\sin(2f\pi t) + 1.0}{2}\right),$$
$$v = U_\infty \sin\left(\frac{\alpha_0 \pi}{180} \times \frac{\sin(2f\pi t) + 1.0}{2}\right),$$

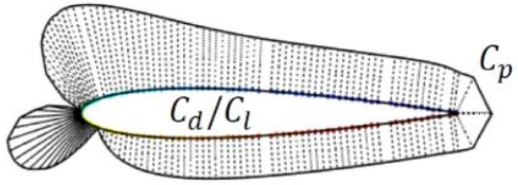
$\alpha = 15^o$
$f = 0.2$
$U_\infty = 1$
$Re = 2500$

- Generate time-dependent AOA



predict

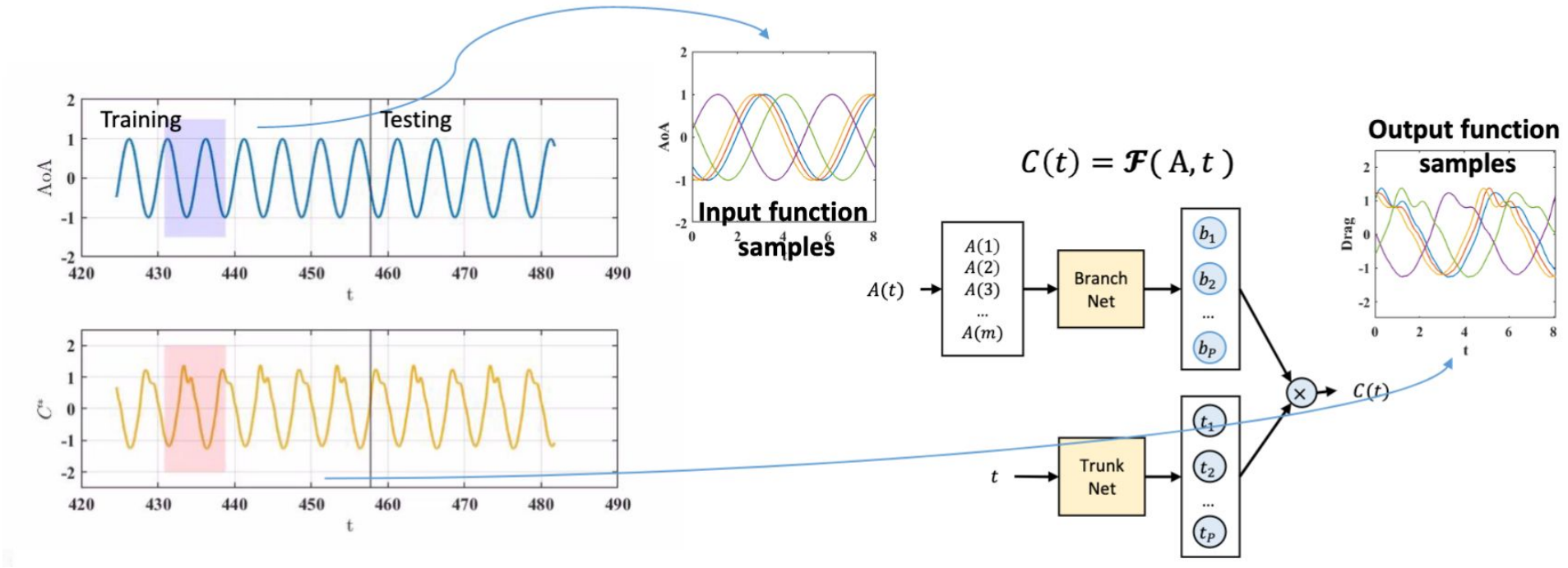
- Time-dependent coefficients of drag, lift, pressure



Applications of Neural Operators

DeepONet for Approximating Functionals: Predicting unsteady pressure and lift/drag-force coefficients

- Predicting drag/lift coefficient
- Cut the time-dependent signal into pieces, to generate a bunch of input-output pairs



Applications of Neural Operators

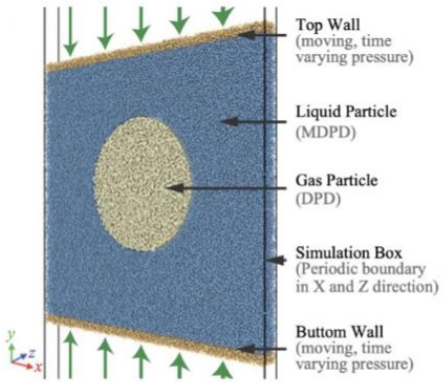
DeepONet for Bubble Dynamics

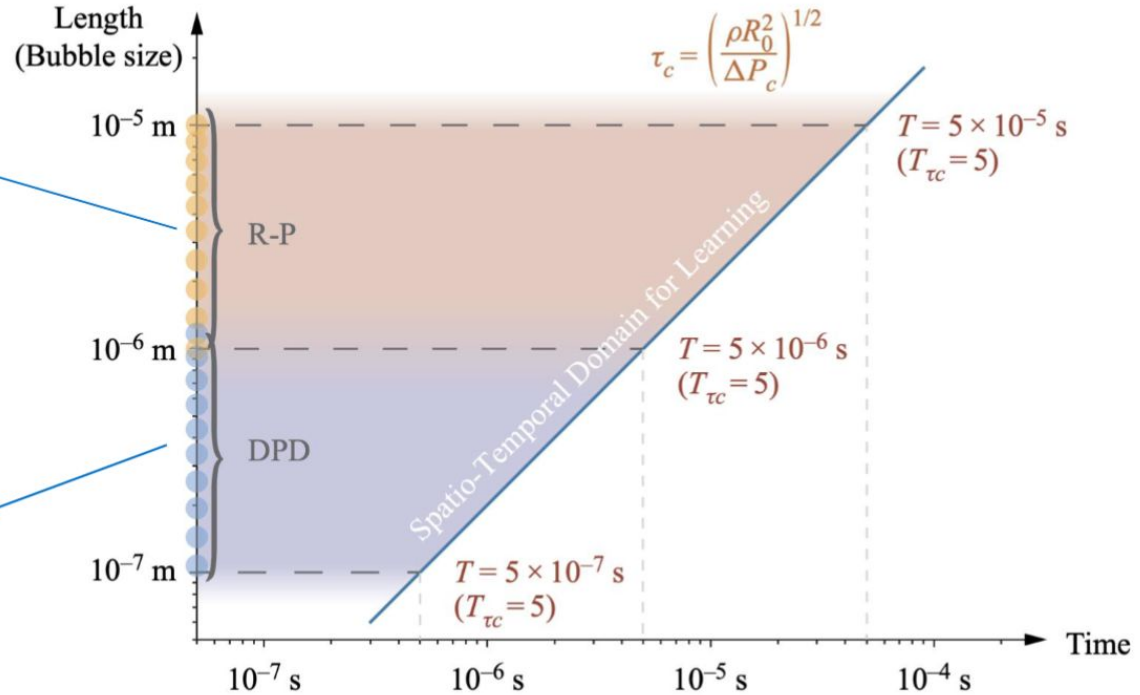
- Rayleigh-Plesset equation is an ordinary differential equation which governs the dynamics of a spherical bubble in an infinite body of imcomporessible fluid
- For nanobubbles, the thermal fluctuation cannot be ignored, and training data are generated by particle simulation

R-P Model

$$\frac{P_\infty(t) - P_B(t)}{\rho_L} = R\frac{d^2R}{dt^2} + \frac{3}{2}\left(\frac{dR}{dt}\right)^2 + \frac{4\nu_L}{R}\frac{dR}{dt} + \frac{2\gamma}{\rho_L R}$$

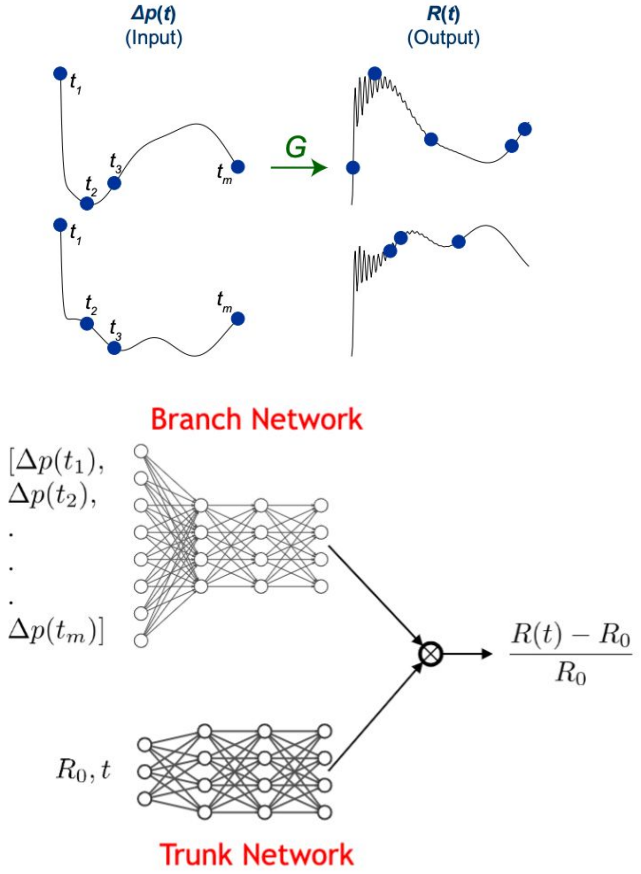
DPD Model

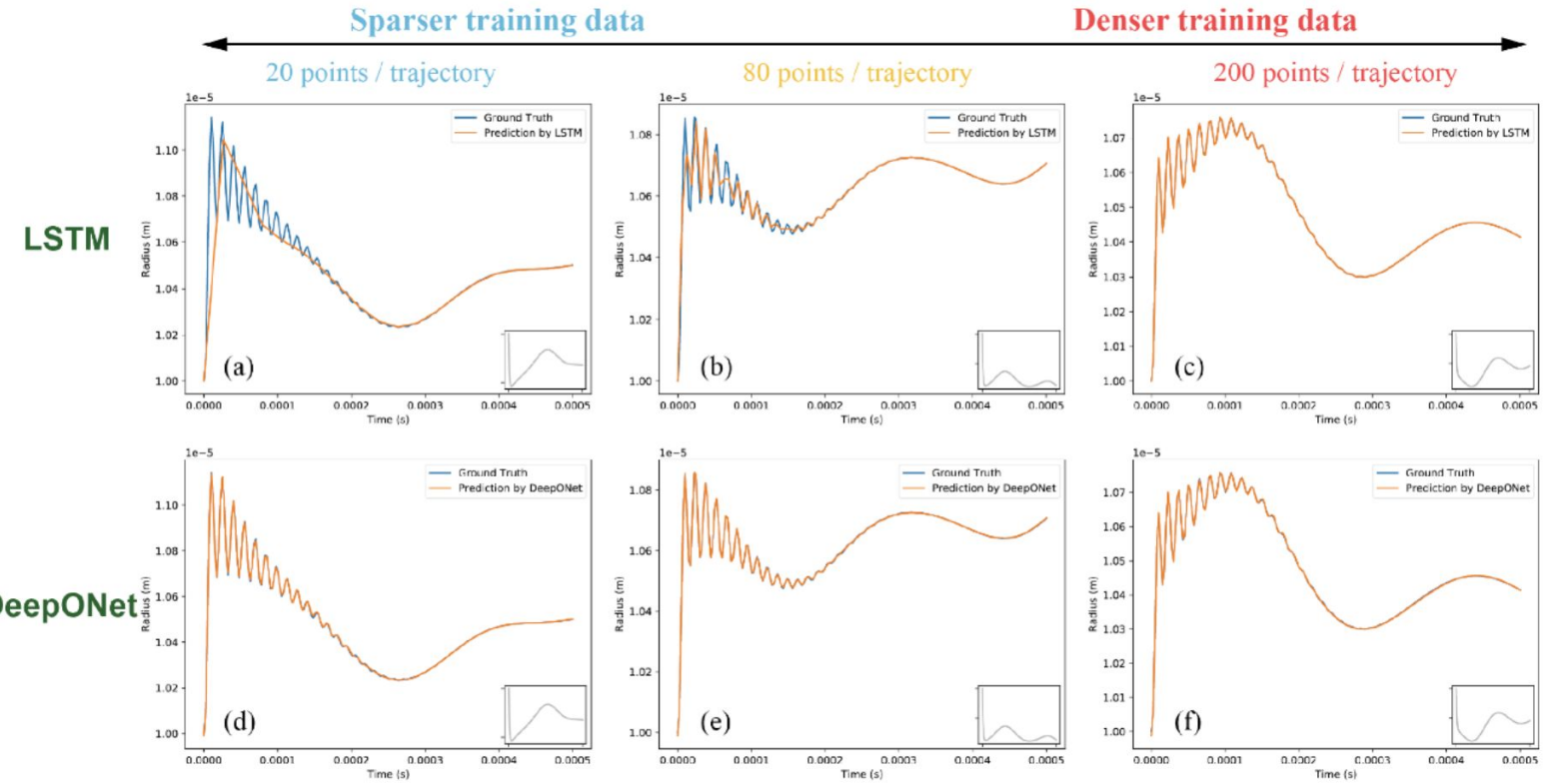




Applications of Neural Operators

DeepONet for Bubble Dynamics





References

• Cai S, Wang Z, Lu L, Zaki TA, Karniadakis GE. DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. Journal of Computational Physics. 2021 Jul 1;436:110296.
• Goswami S, Yin M, Yu Y, Karniadakis GE. A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. Computer Methods in Applied Mechanics and Engineering. 2022 Mar 1;391:114587.
• Lanthaler S, Mishra S, Karniadakis GE. Error estimates for deeponets: A deep learning framework in infinite dimensions. arXiv preprint arXiv:2102.09618. 2021 Feb 18.
• Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, Anandkumar A. Fourier neural operator for parametric partial differential equations. arXiv preprint arXiv:2010.08895. 2020 Oct 18.
• Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Machine Intelligence. 2021 Mar;3(3):218-29.
• Lu L, Meng X, Cai S, Mao Z, Goswami S, Zhang Z, Karniadakis GE. A comprehensive and fair comparison of two neural operators (with practical extensions) based on FAIR data. arXiv preprint arXiv:2111.05512. 2021 Nov 10.
• Mao Z, Lu L, Marxen O, Zaki TA, Karniadakis GE. DeepM&Mnet for hypersonics: Predicting the coupled flow and finite-rate chemistry behind a normal shock using neural-network approximation of operators. Journal of Computational Physics. 2021 Dec 15;447:110698.

THANK YOU