February 18, 2025

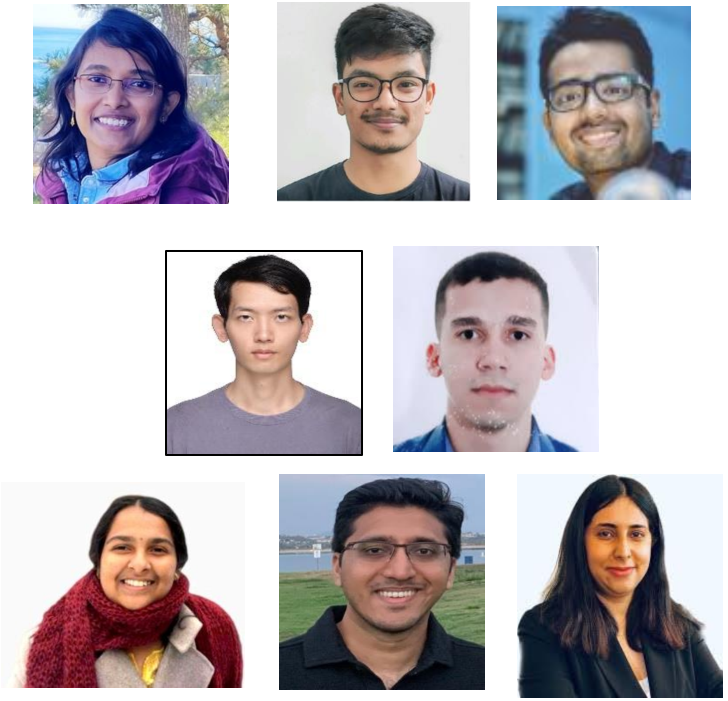
Scalable Surrogate Models for High-Dimensional Physics-based Systems

Somdatta Goswami
Civil and Systems Engineering
Johns Hopkins University

BIRS workshop Uncertainty Quantification in Neural Network Models

Centrum IntelliPhysics

- Mission: Develop machine learning tools to accelerate engineering innovation
- Focus: Physics-Informed Machine Learning
  - Efficient training strategies for neural operators
  - Developing hybrid solvers (operators + solvers)
- Applications: Multiscale Modeling in Materials, Engineering and Biomedical Systems



JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Physics-based Models

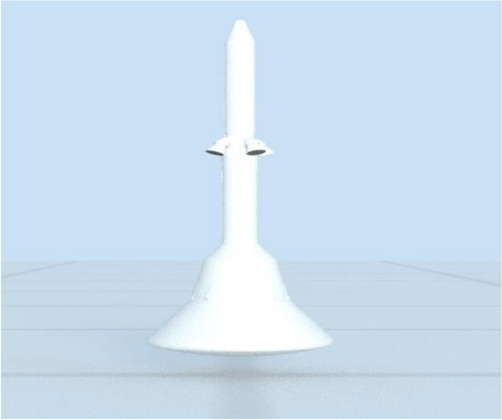
Can represent the **Processes of Nature**

- Physics-based models are approximated via **ODEs/PDEs**

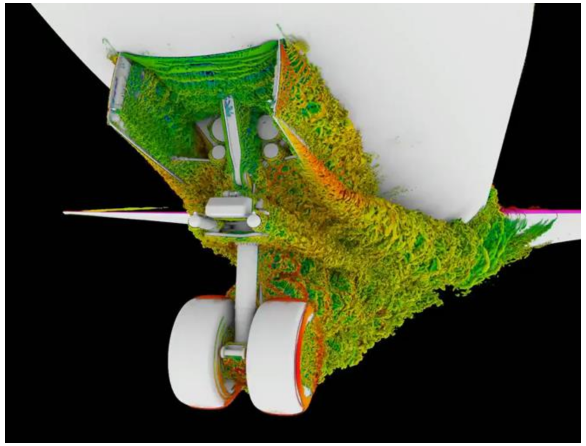
  To model earthquake: $m\frac{d^2u}{dt^2} + \mathrm{k}\frac{du}{dt} + F_0 = 0$

  To model waves: $\frac{\partial^2 u}{\partial t^2} - v^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0$

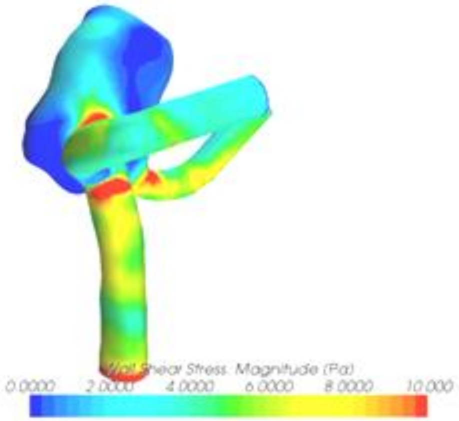
- Computational Mechanics helps us simulate these equations.



Simulation of Orion Spacecraft Launch Abort System (NASA Ames)



Detailed flow around an Aircraft's landing gear (NASA Ames)



CFD Simulation of a Patient-Specific Intracranial Aneurysm

Challenges with Numerical Methods

- Require knowledge of conservation laws, and boundary conditions
- Time consuming and strenuous simulations.
- Difficulties in mesh generation.
- Solving inverse problems or discovering missing physics can be prohibitively expensive.

Develop Physics-based surrogate models for these systems to create a fast-to-evaluate alternative.

Surrogate Modeling Techniques

- Discretized Data
- Discretization dependent
- Queries on mesh
- Learning functions between vector spaces

- Functional Data
- Discretization Invariant
- Continuous quantities
- Learning operators between function spaces

Data-driven

PCA **Auto-encoders**

K-PCA **Diffusion maps**

f-PCA **DeepONet** **LNO**

FNO **WNO**

F-RKHS

Finite Dimensional

Infinite Dimensional

Physics Informed

PINNs

PI-DeepONet

PINO

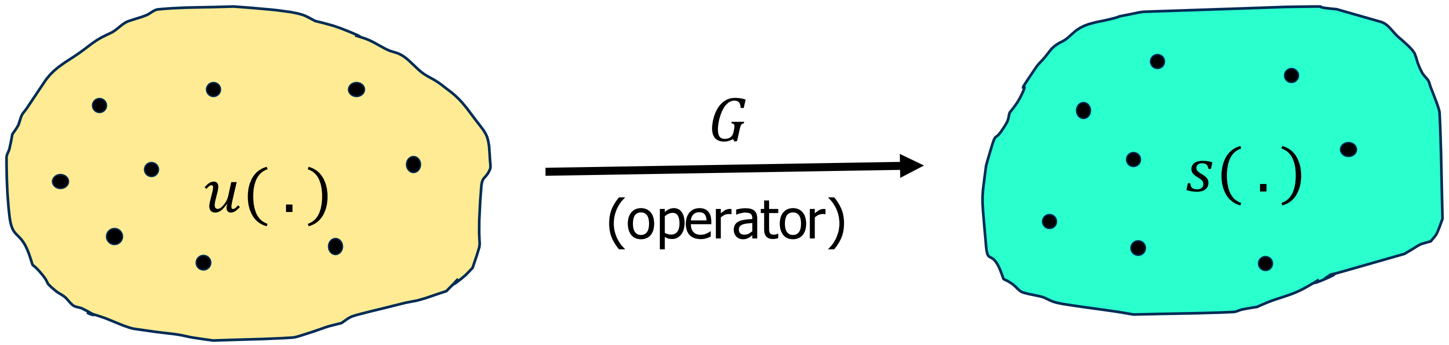
Operator Learning Framework

Input-output map

$$\Phi: \mathcal{U} \rightarrow \mathcal{S}$$

Data $\{\mathcal{U}_n, \mathcal{S}_n\}_{n=1}^{N}$and/or Physics

$\mathcal{S}_n = \Phi(\mathcal{F}_n)$, $\mathcal{F}_n \sim \mu \; i.i.d$



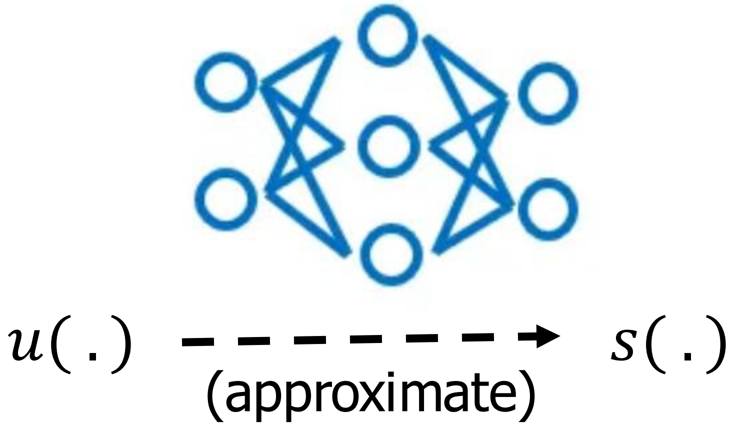
Input function space

Output function space

Operator learning

$$\Psi{:}\times\, \Theta \rightarrow \mathcal{S} \text{ such that } \Psi(.,\theta^*) \approx \Phi$$

Training $\theta^* = \mathrm{argmin}_\theta\, l(\{\mathcal{U}_n, \Psi(\mathcal{S}_n, \theta)\})$

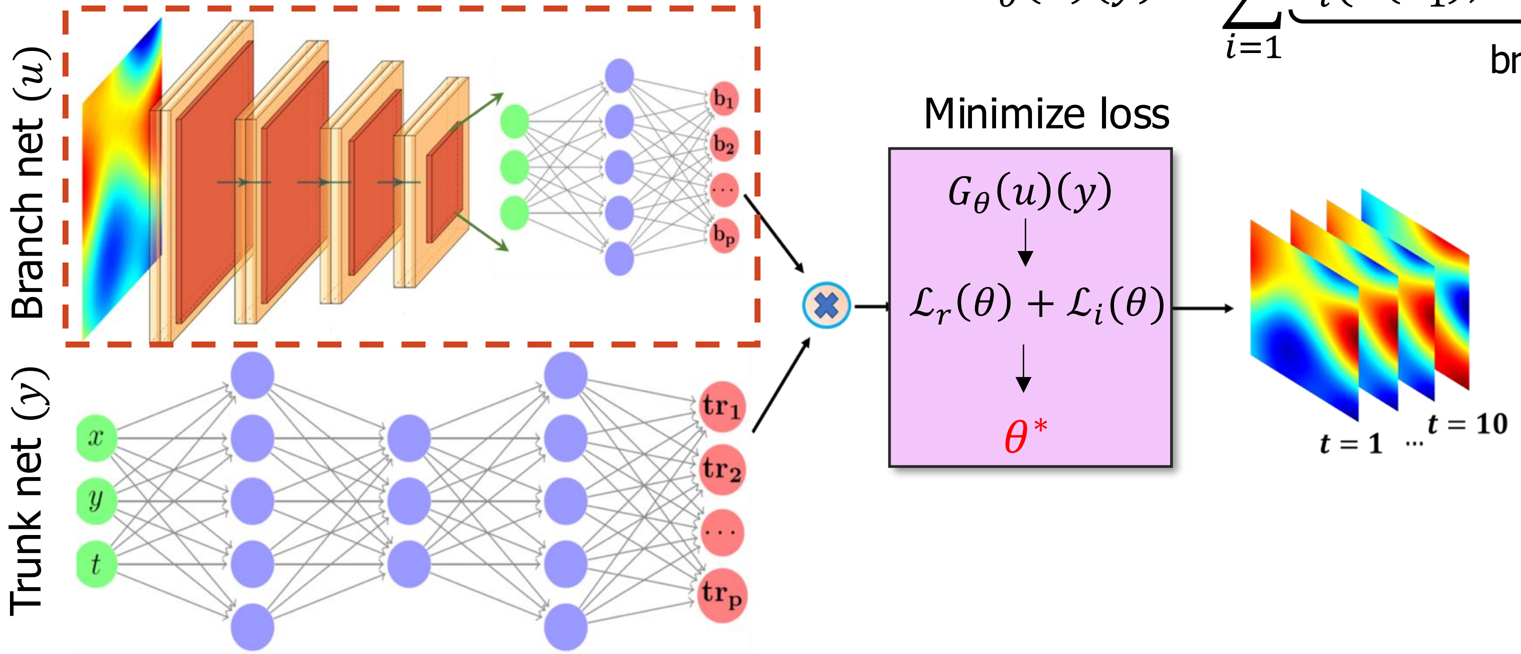


Deep Operator Network (DeepONet)

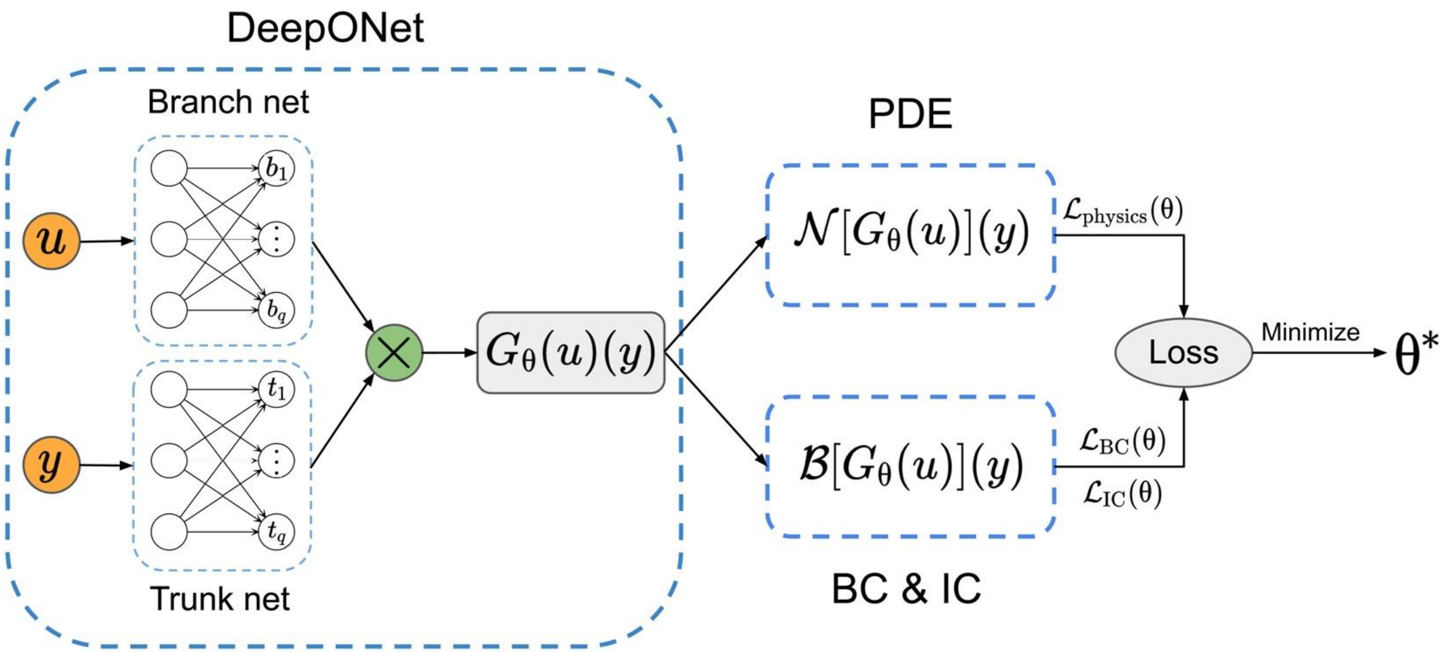
- Generalized Universal Approximation Theorem for Operator [Chen ’95, Lu et al. ’19]
- **Branch net**: Input $\{u(x_i)\}_{i=1}^{m}$, output: $\left[b_1, b_2, .., b_p\right]^T \in \mathbb{R}^p$
- **Trunk net**: Input y, output: $\left[t_1, t_2, .., t_p\right]^T \in \mathbb{R}^p$
- Input u is evaluated at the fixed locations $\{y_i\}_{i=1}^{m}$

$$G_\theta(u)(y) = \sum_{i=1}^{p} \underbrace{b_i(u(x_1), u\,(x_2), \dots, u\,(x_m))}_{\text{branch net}} \cdot \underbrace{tr_i(y)}_{\text{trunk net}}$$

Can be FNN/CNN/U-Net/LSTM etc.



Physics-Informed DeepONet



- Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Science Advances, 7(40), October 2021.
- Somdatta Goswami, Yin, M., Yu, Y., & Karniadakis, G. E. (2022). A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. Computer Methods in Applied Mechanics and Engineering, 391, 114587.

Our Proposed framework

ELSEVIER

Computer Methods in Applied Mechanics and Engineering

Volume 434, 1 February 2025, 117586

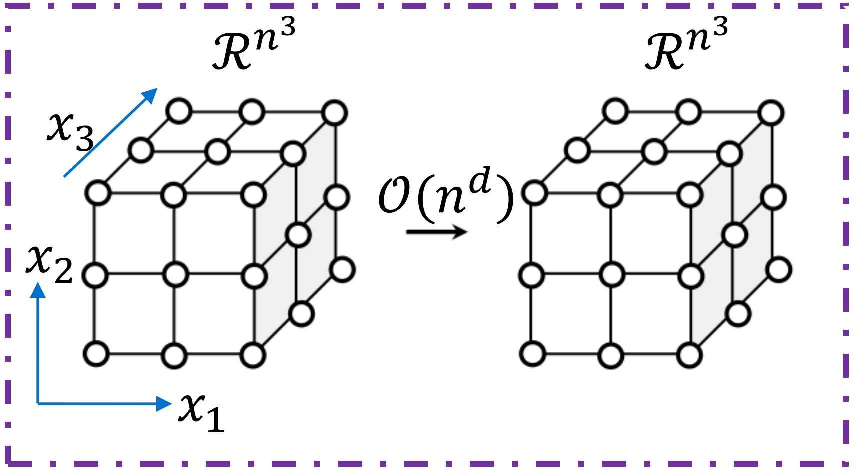
Separable physics-informed DeepONet: Breaking the curse of dimensionality in physics-informed machine learning

Luis Mandl [a], Somdatta Goswami [b], Lena Lambers [a], Tim Ricken [a]

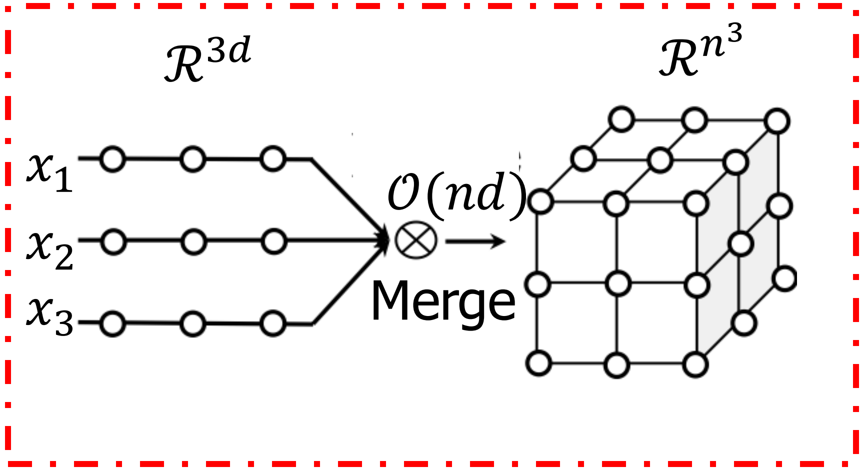



Introducing Separation of Variables

Vanilla Trunk network

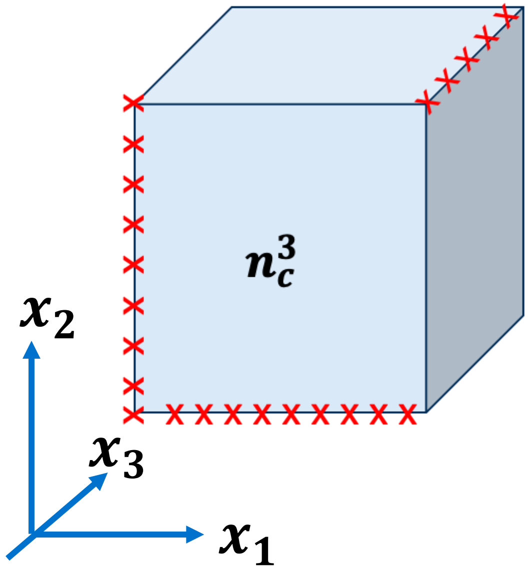


Separated Trunk network

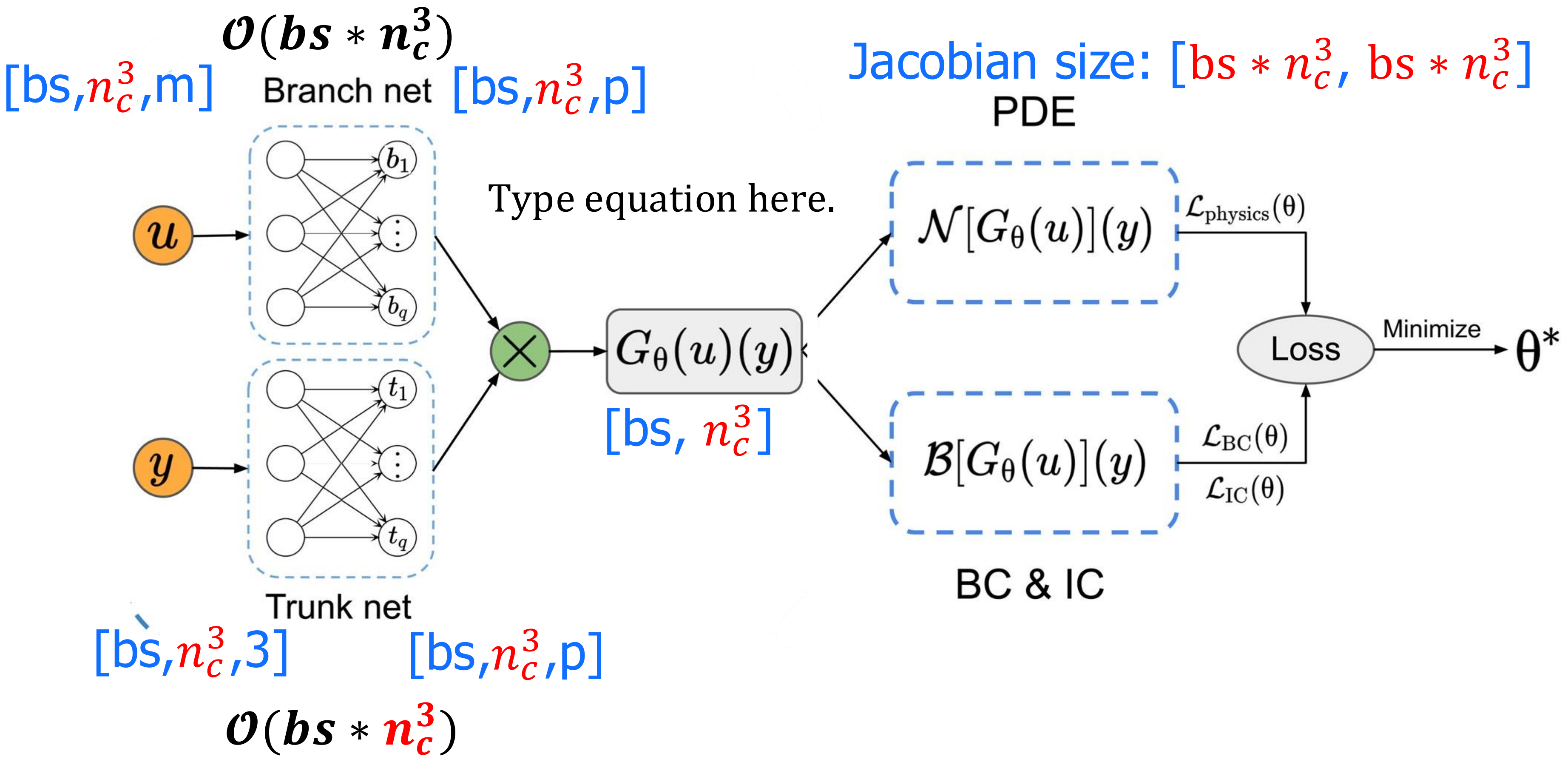


Introduced in PINNs : Cho, J., Nam, S., Yang, H., Yun, S. B., Hong, Y., & Park, E. (2022). Separable PINN: Mitigating the curse of dimensionality in physics-informed neural networks. ArXiv preprint - 2211.08761.

Vanilla – Physics Informed DeepONet

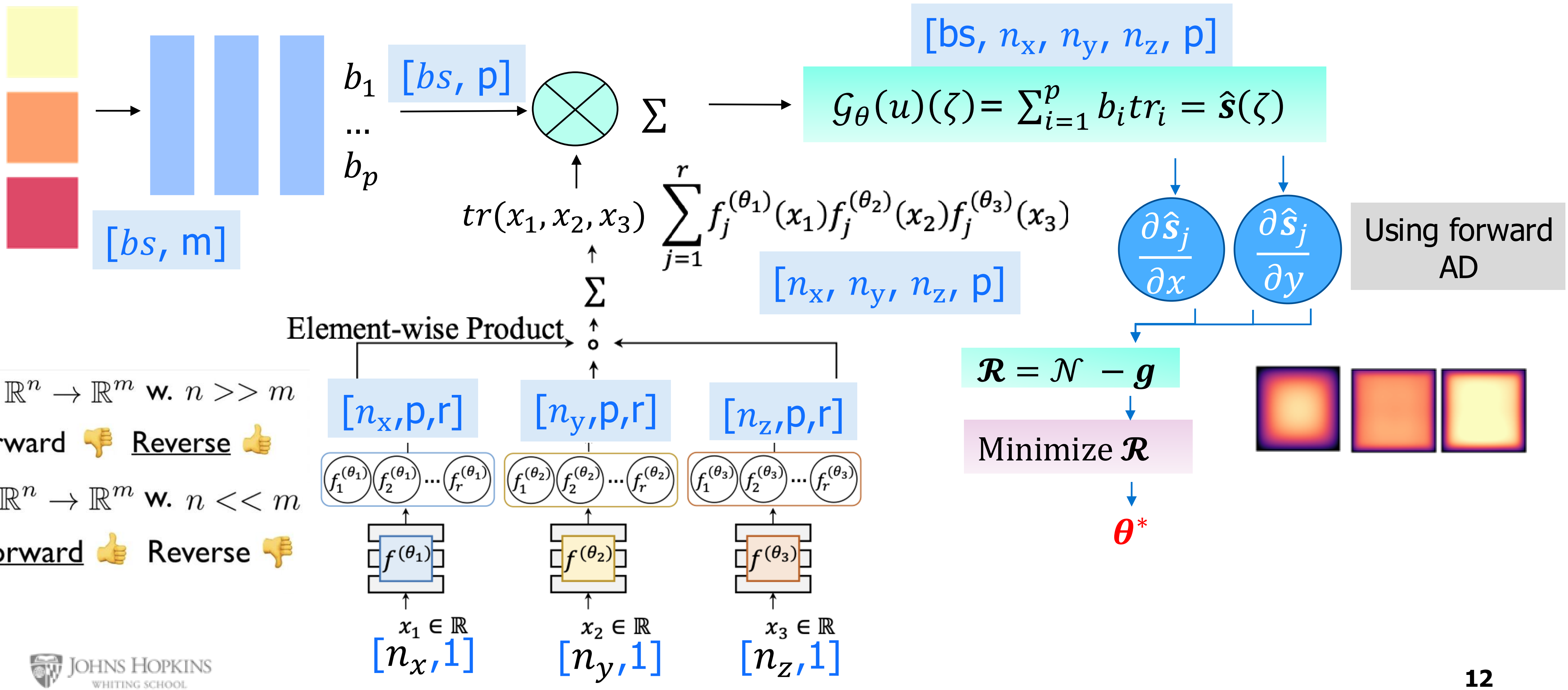


$n_{x_1} = n_{x_2} = n_{x_3} = n_c$



Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed DeepONets. Science Advances, 7(40), October 2021.

Separable DeepONet Framework



$f : \mathbb{R}^n \to \mathbb{R}^m$ w. $n >> m$

Forward 👎 Reverse 👍

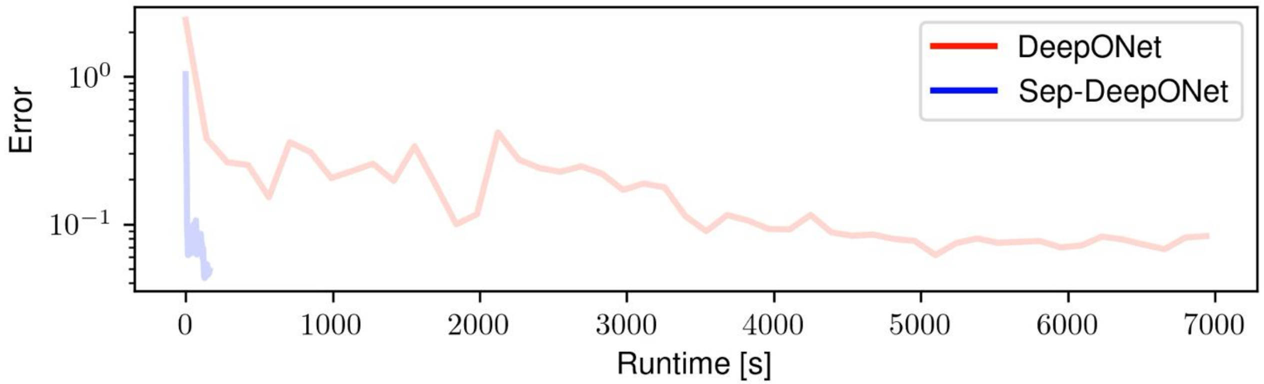
$f : \mathbb{R}^n \to \mathbb{R}^m$ w. $n << m$

Forward 👍 Reverse 👎

Numerical Examples

Problem	Model	d	Relative $\mathcal{L}_2$ error	Run-time (ms/iter.)
Burgers Equation	Vanilla Separable (Ours)	2	$5.1e\text{-}2$ $6.2e\text{-}2$	136.6 3.64
Consolidation Biot's Theory	Vanilla Separable (Ours)	2	$7.7e\text{-}2$ $7.9e\text{-}2$	169.43 3.68
Parameterized Heat Equation	Vanilla Separable (Ours)	4	- $7.7e\text{-}2$	10,416.7 91.73

Burgers’ Equation

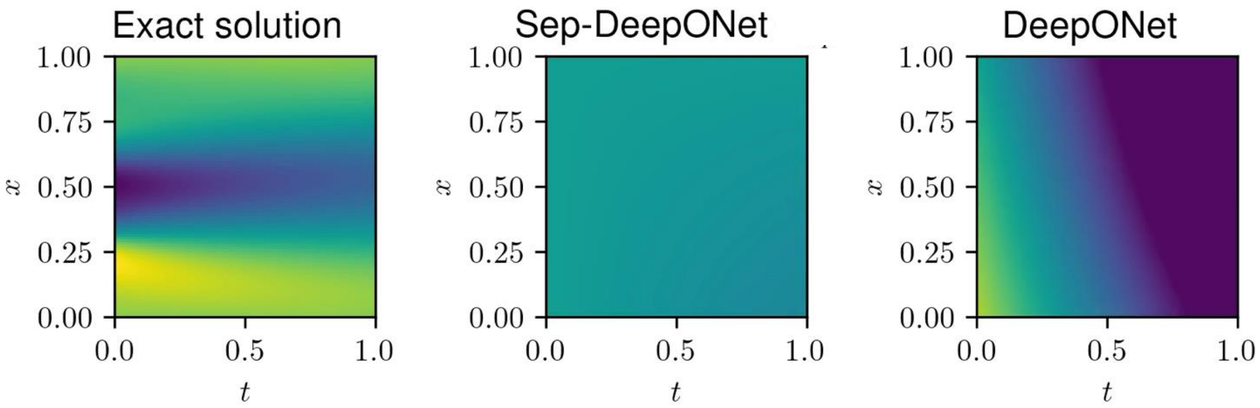


$$\frac{\partial s(x,t)}{\partial t} + s\frac{\partial s(x,t)}{\partial x} - \nu\frac{\partial^2 s(x,t)}{\partial x^2} = 0$$

$$s(0,t) = s(1,t),$$

$$\frac{\partial s(0,t)}{\partial x} = \frac{\partial s(1,t)}{\partial x},$$

$$s(x,0) = u(x), \quad x \in [0,1]$$



Model	Branch	Trunk	p	r	Parameters	$\mathcal{L}_2$ rel. err.	Runtime [s]	Runtime improvment
Vanilla PI-DeepONet	6×[100]	6×[100]	100	-	131,701	5.14e-2	6,829.2	-
Sep-PI-DeepONet	6×[100]	6×[100]	50	50	672,151	6.24e-2	182.1	97,33%
	6×[100]	6×[100]	20	20	244,921	6.04e-2	197.8	97,10%
	6×[100]	6×[50]	20	20	129,221	6.46e-2	197.0	97,12%

Biot’s Consolidation

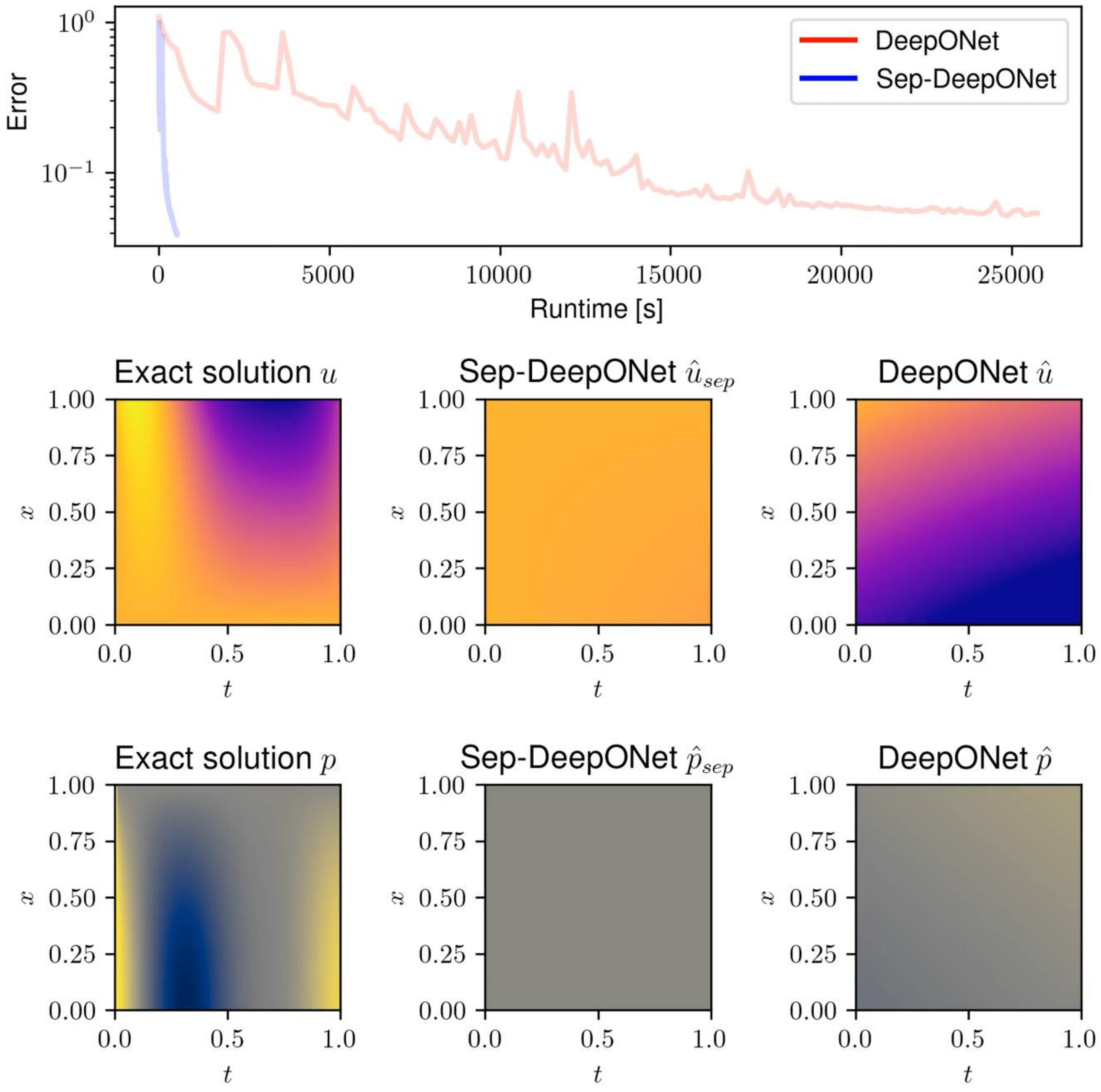
$$(\lambda + 2\mu)\frac{\partial^2 u(z,t)}{\partial z^2} - \frac{\partial p(z,t)}{\partial z} = 0$$

$$\frac{\partial^2 u(z,t)}{\partial t \partial z} - \frac{k}{\rho g}\frac{\partial^2 \tilde{p}(z,t)}{\partial z^2} = 0,$$

$$u(z,0) = 0, \qquad p(0,t) = 0,$$

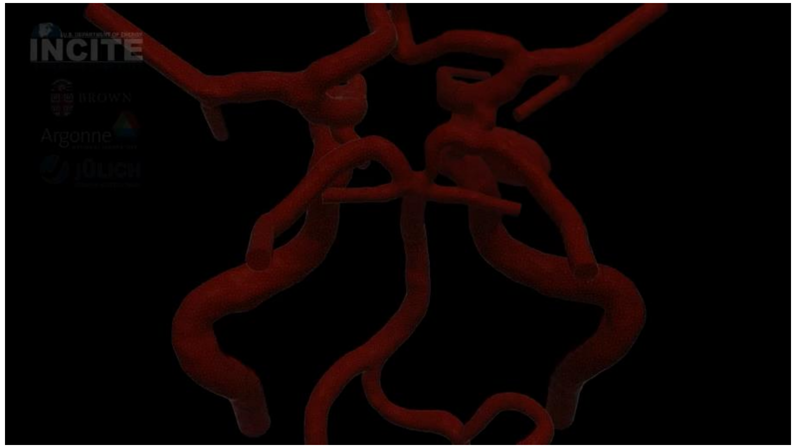
$$p(z,0) = f(0), \qquad u(L,t) = 0,$$

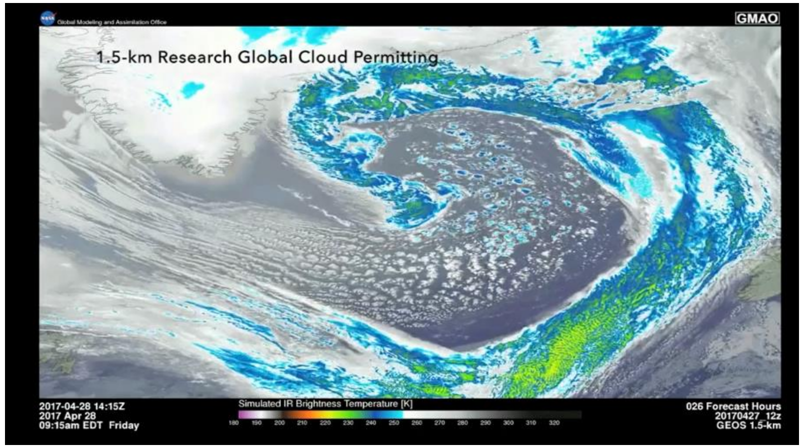
$$\sigma(0,t) = -f(t), \qquad \frac{\partial p(L,t)}{\partial z} = 0,$$

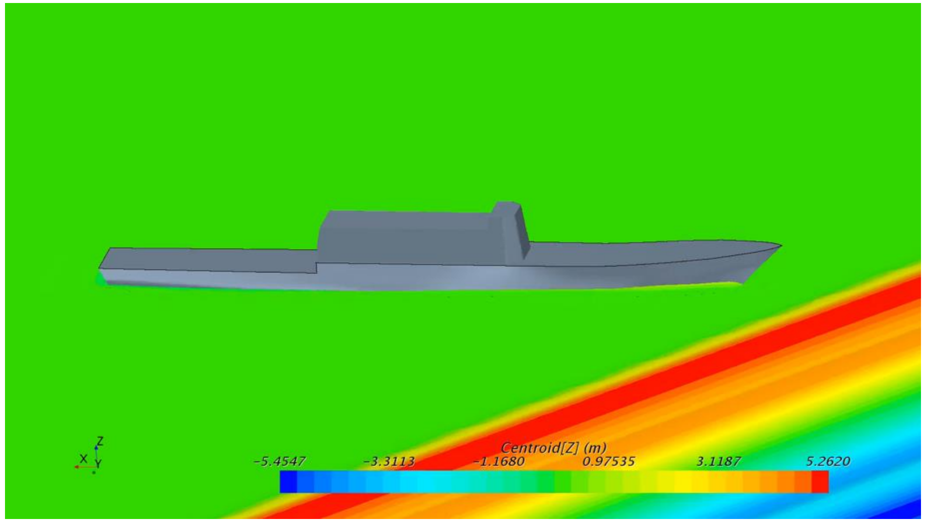


Implementing Physics-Informed DeepONet is not an easy task for complicated systems

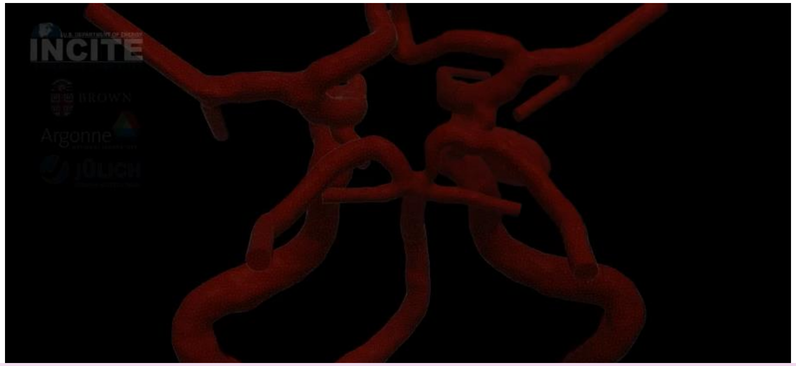
Can we harness the explosion of data to extract knowledge, insight and decision?

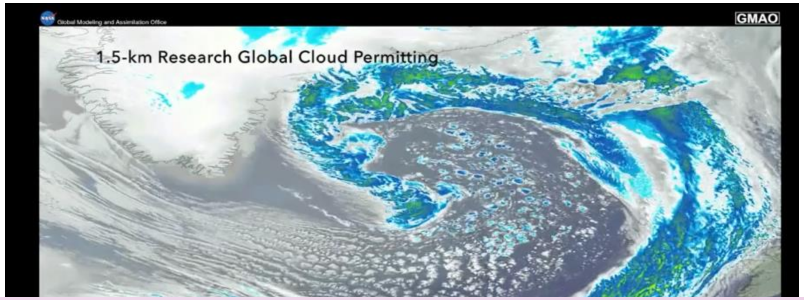

INCITE
Argonne


GMAO
1.5-km Research Global Cloud Permitting
2017-04-28 14:15Z
2017 Apr 28
09:15am EDT Friday
Simulated IR Brightness Temperature [K]
026 Forecast Hours
20170427_12z
GEOS 1.5-km


Centroid[Z] (m)
-5.4547 -3.3113 -1.1680 0.97535 3.1187 5.2620

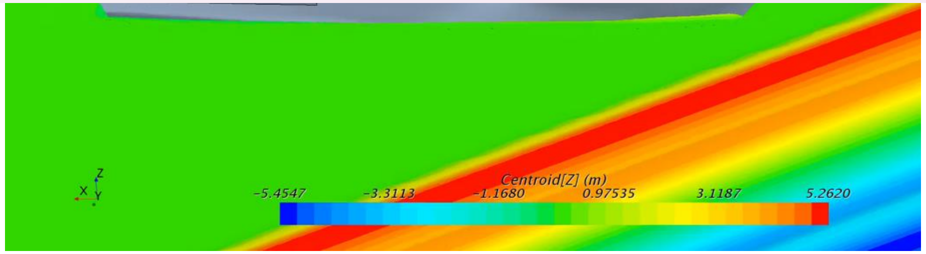

StreamTime LIVE - Baltimore, MD 2024-03-26 01:28:45 EDT





BIG Decisions need BIG MODELS

But we have: sparse high-dimensional datasets



Outline

Physics-Informed Operator Learning on Latent Spaces

Part – I: Data-driven operator learning on reduced spaces

Part – II: Integrating physics and data to learn operator on reduced spaces

Our Proposed framework

nature communications

Article

https://doi.org/10.1038/s41467-024-49411-w

Learning nonlinear operators in latent spaces for real-time predictions of complex dynamics in physical systems




Viscous Shallow water equation

- Model the dynamics of large-scale atmospheric flows
- Perturbation is used to induce the development of barotropic instability

$$\frac{\mathrm{d}\boldsymbol{V}}{\mathrm{d}t} = -f\boldsymbol{k} \times \boldsymbol{V} - g\nabla h + \nu\nabla^2\boldsymbol{V}$$

$$\frac{\mathrm{d}h}{\mathrm{d}t} = -h\nabla \cdot \boldsymbol{V} + \nu\nabla^2 h$$

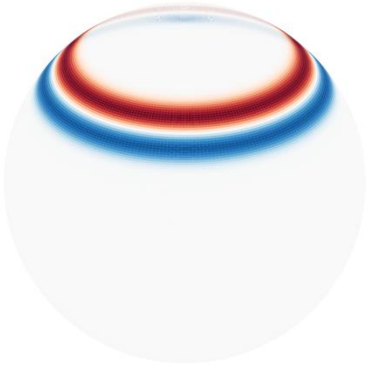
$$h'(\lambda, \phi) = \hat{h}\cos(\phi)\mathrm{e}^{-(\lambda/\alpha)^2}\mathrm{e}^{-[(\phi_2-\phi)/\beta]^2}$$

rvs: $\alpha \sim U[0.\bar{1}, 0.5]$ $\beta \sim U[0.0\bar{3}, 0.2]$

Operator: $\mathcal{G}: h'(\lambda, \varphi, t = 0) \longmapsto u(\varphi, \lambda, t)$

Input Dimension: 65,536

Gaussian Random Perturbation



Output Dimension: 4,718,592

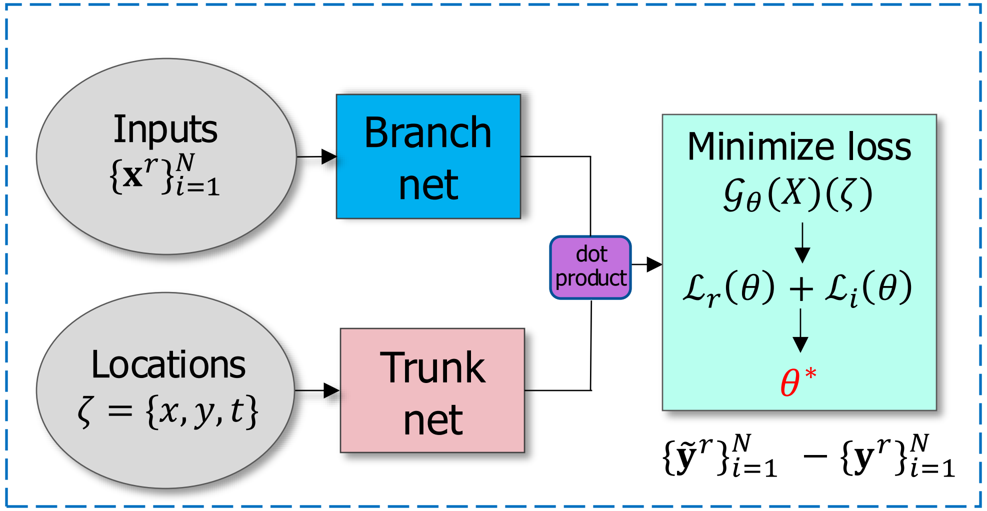
Atmospheric Flow

Latent DeepONet for time-dependent PDEs

Operator $\mathcal{G}: \mathcal{X} \to \mathcal{Y}$

$\mathcal{G}_\theta: \mathcal{X} \to \mathcal{Y}, \quad \theta \in \Theta$

Training data $\{\mathbf{x}_i, \boldsymbol{y}_i\}_{i=1}^N$

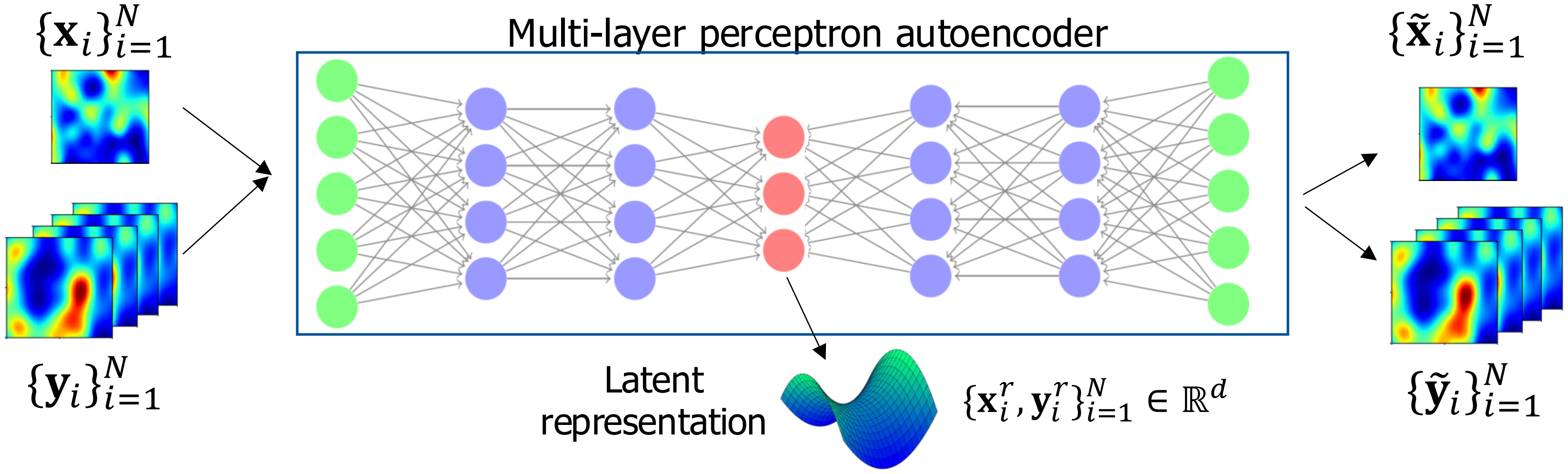


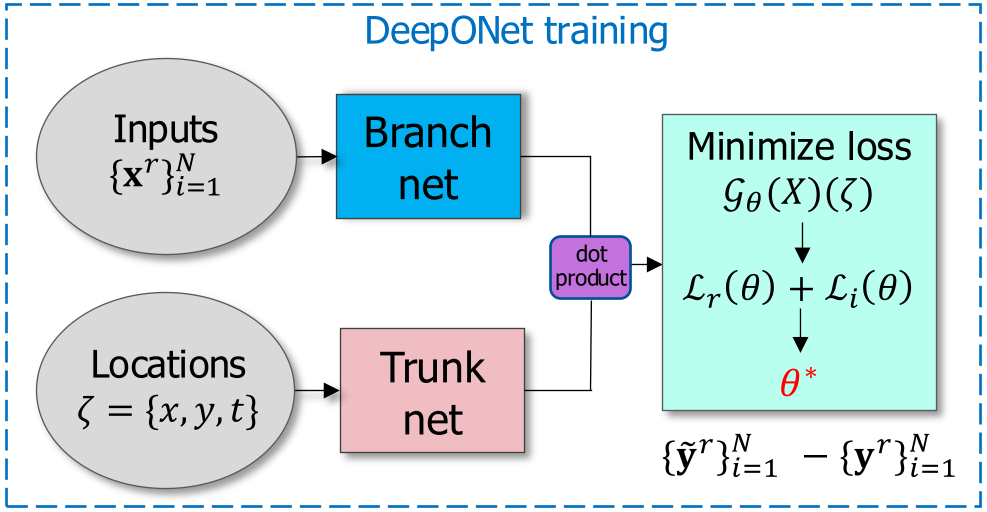
Latent DeepONet for time-dependent PDEs

Operator $\mathcal{G} : \mathcal{X} \to \mathcal{Y}$

$\mathcal{G}_\theta : \mathcal{X} \to \mathcal{Y}, \quad \theta \in \Theta$

Training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$



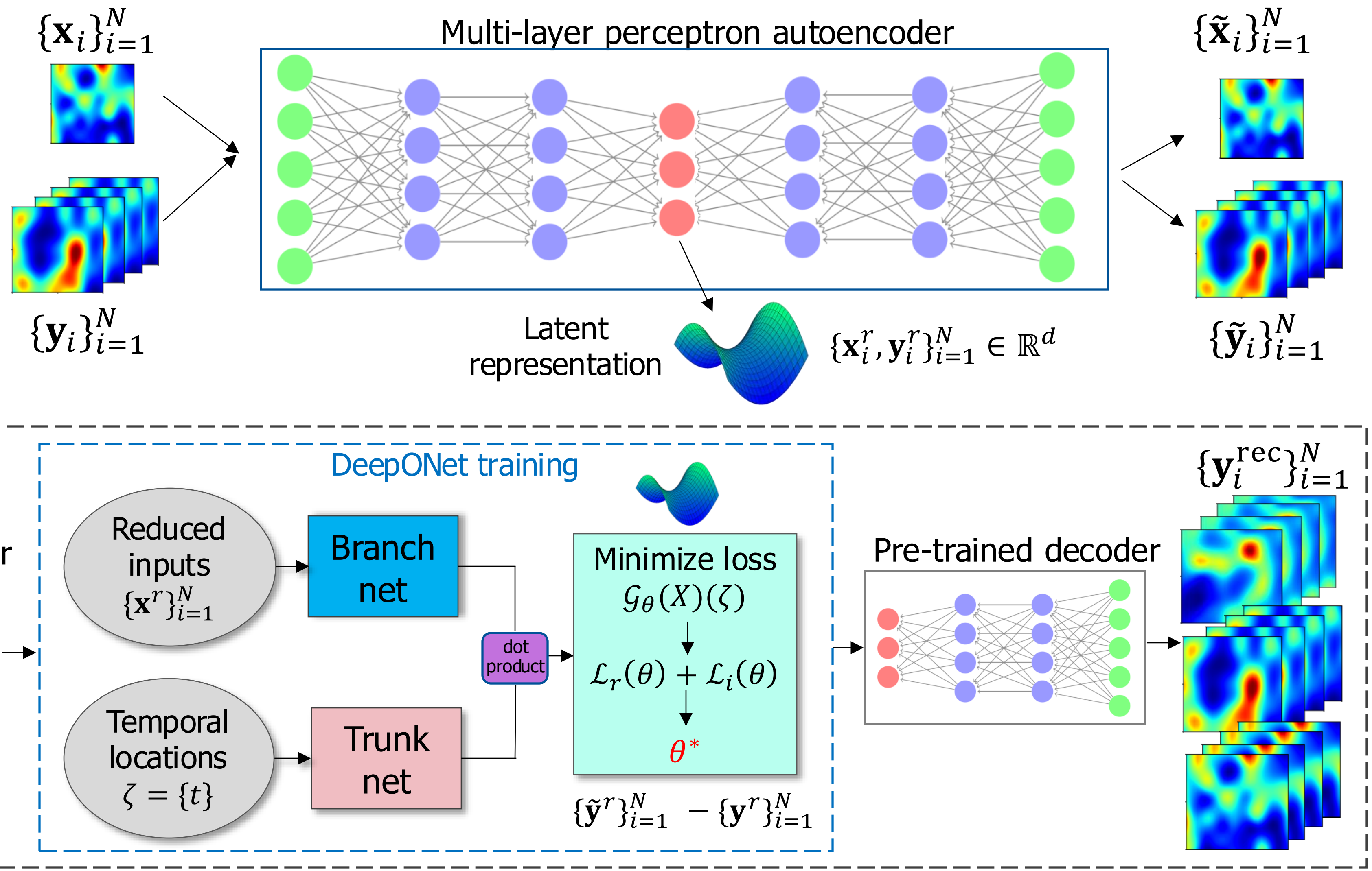


Latent DeepONet for time-dependent PDEs

Operator $\mathcal{G}: \mathcal{X} \to \mathcal{Y}$

$\mathcal{G}_\theta: \mathcal{X} \to \mathcal{Y}, \quad \theta \in \Theta$

Training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$



MLAE: multi-layer autoencoders
PCA: principal component analysis

Consolidated results

Accuracy of *L-DeepONet for MLAE and PCA*

Application	d	with MLAE	with PCA
Brittle material fracture	9	$3.33 \cdot 10^{-4} \pm 4.99 \cdot 10^{-5}$	$2.71 \cdot 10^{-3} \pm 6.62 \cdot 10^{-6}$
	64	$2.02 \cdot 10^{-4} \pm 1.88 \cdot 10^{-5}$	$3.13 \cdot 10^{-4} \pm 4.62 \cdot 10^{-6}$
Rayleigh-Bénard fluid flow	25	$4.10 \cdot 10^{-3} \pm 8.05 \cdot 10^{-5}$	$3.90 \cdot 10^{-3} \pm 4.73 \cdot 10^{-5}$
	100	$3.55 \cdot 10^{-3} \pm 1.46 \cdot 10^{-4}$	$3.76 \cdot 10^{-3} \pm 4.86 \cdot 10^{-5}$
Shallow water equation	25	$2.30 \cdot 10^{-4} \pm 1.50 \cdot 10^{-5}$	$7.98 \cdot 10^{-4} \pm 8.01 \cdot 10^{-7}$
	81	$2.23 \cdot 10^{-4} \pm 1.83 \cdot 10^{-5}$	$4.18 \cdot 10^{-4} \pm 4.67 \cdot 10^{-6}$

Computational training time in seconds (s) on an NVIDIA A6000 GPU

Application	L-DeepONet	Full DeepONet	FNO-3D
Brittle material fracture	1,660	15,031	128,000
Rayleigh-Bénard fluid flow	2,853	6,772	1,126,400
Shallow water equation	15,218	379,022	–

Spherical shallow water equations

- Model the dynamics of large-scale atmospheric flows
- Barotropically unstable mid-latitude jet (*Ref: Galewsky et al. 2004*)
- Perturbation is used to induce the development of barotropic instability

Shallow-water equations

$$\frac{\mathrm{d}\boldsymbol{V}}{\mathrm{d}t} = -f\boldsymbol{k} \times \boldsymbol{V} - g\nabla h + \nu\nabla^2\boldsymbol{V}$$

$$\frac{\mathrm{d}h}{\mathrm{d}t} = -h\nabla \cdot \boldsymbol{V} + \nu\nabla^2 h$$

- $\boldsymbol{V} = iu + jv$: velocity vector tangent to the sphere
- h: height field (thickness of the fluid layer)
- $f = 2\Omega\sin\varphi$: Coriolis parameter
- φ: latitude, Ω: angular velocity of Earth, ν: diff. coeff.

Initial condition

$$u(\phi) = \begin{cases} 0 & \text{for} \quad \phi \le \phi_0 \\ \dfrac{u_{\max}}{e_n} \exp\left[\dfrac{1}{(\phi-\phi_0)(\phi-\phi_1)}\right] & \text{for} \quad \phi_0 < \phi < \phi_1 \\ 0 & \text{for} \quad \theta \ge \phi_1 \end{cases}$$

$$h'(\lambda, \phi) = \hat{h}\cos(\phi)\mathrm{e}^{-(\lambda/\alpha)^2}\mathrm{e}^{-[(\phi_2-\phi)/\beta]^2}$$

rvs: $\alpha \sim U[0.\bar{1}, 0.5]$ $\beta \sim U[0.0\bar{3}, 0.2]$

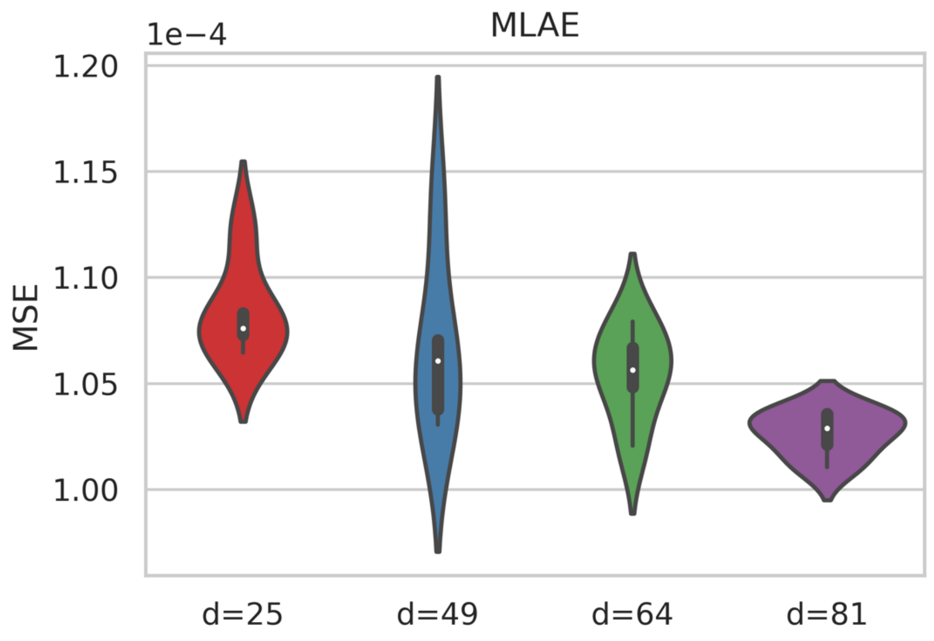
Operator: $\mathcal{G}: h'(\lambda, \varphi, t = 0) \longmapsto u(\varphi, \lambda, t)$

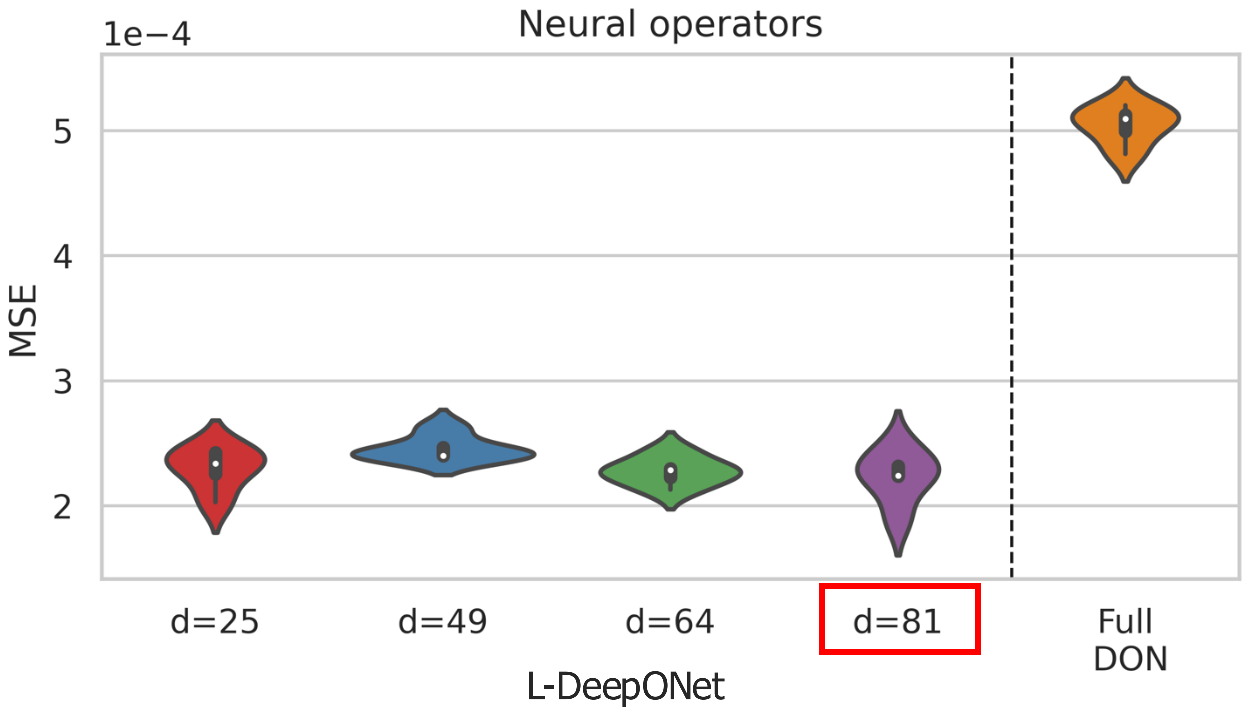
Results

- $\Omega = [0,2\pi]x[0,2\pi]$, $(n_x \times n_y) = (256x256)$ mesh points
- Output dimensionality: $72x256x256 = 4{,}718{,}592$
- Simulation: $t = [0{,}360h], \delta t = 0.1\bar{6}h$, Time steps: $n_t = 72$

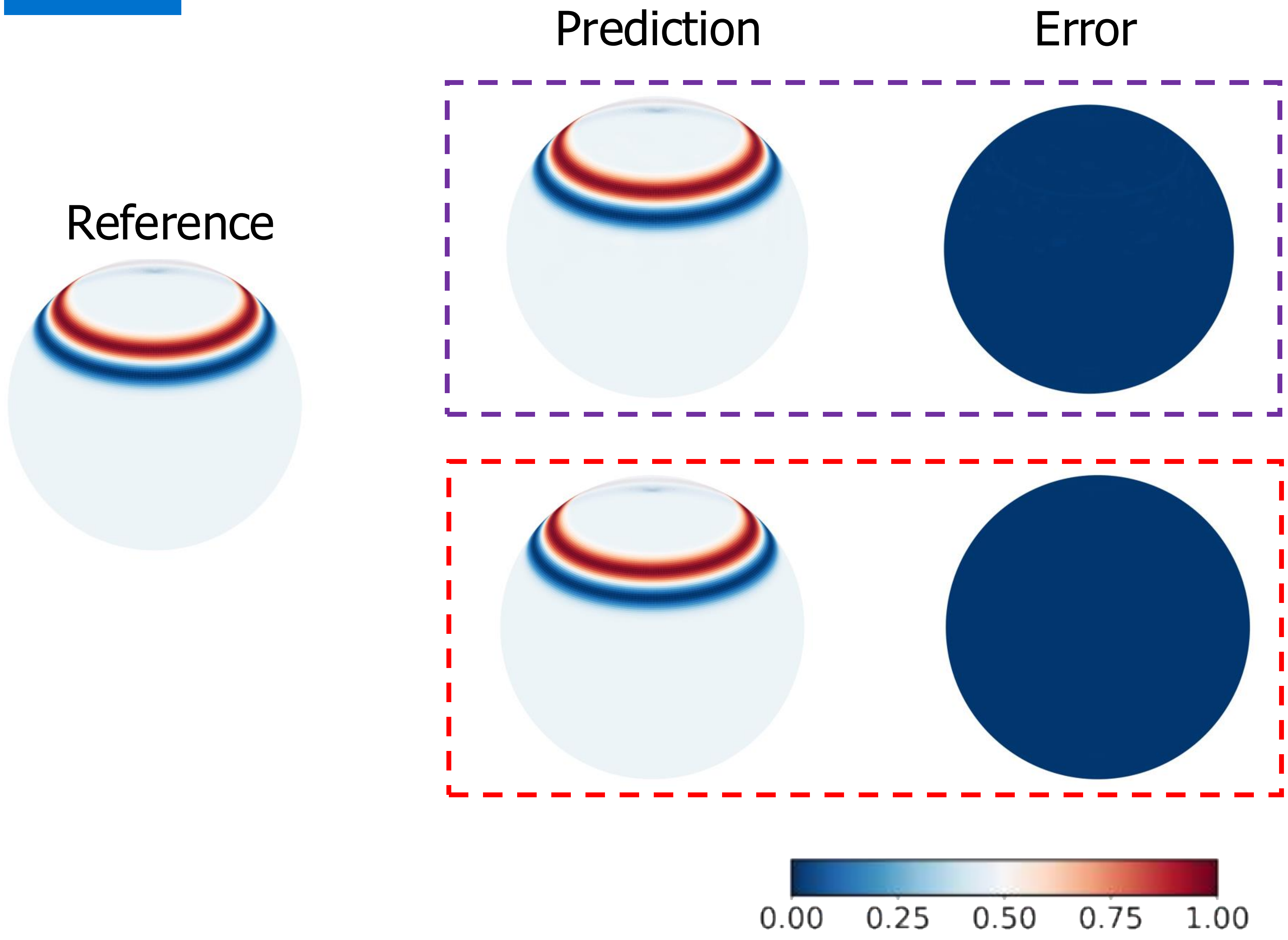
Training Time (seconds)

MLAE + Latent DON: 15, 218
Full DON: 379,022





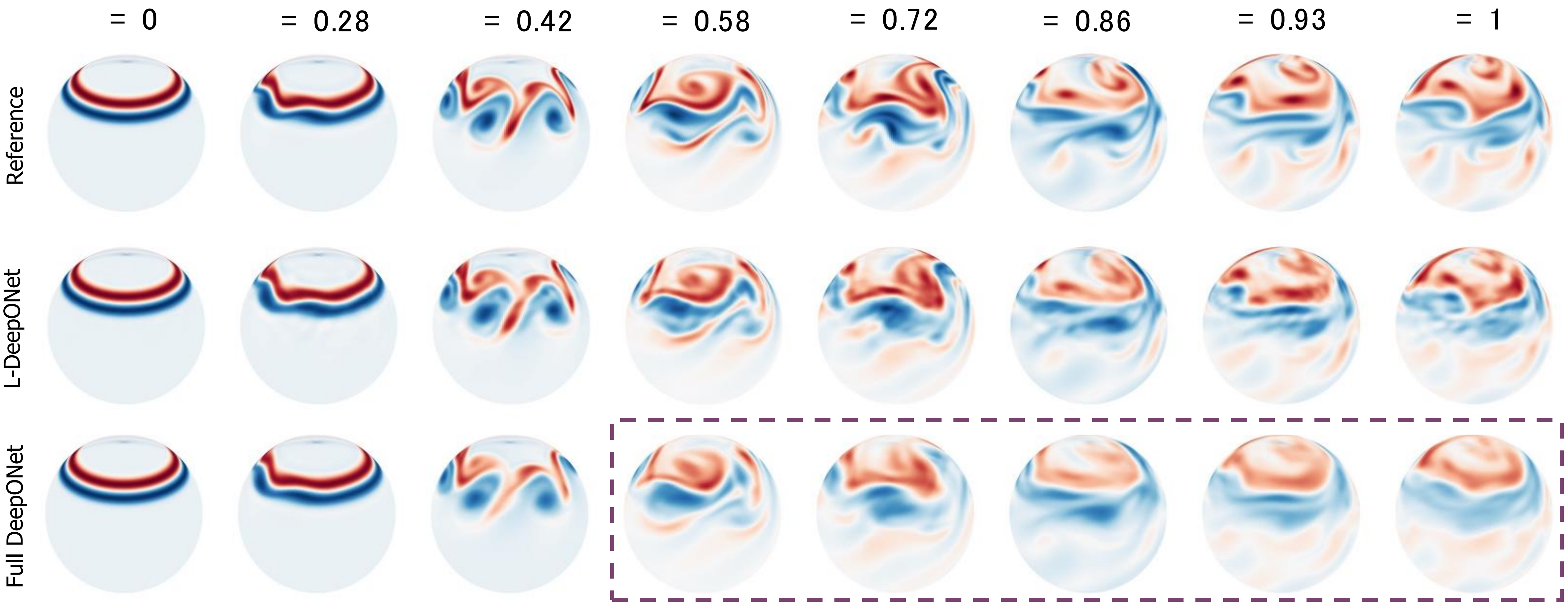
Results



Latent DeepONet
(275,475 Parameters)

DeepONet
(327,872 Parameters)

Latent DeepONet and Full DeepONet



Shortcomings

↑

The framework requires voluminious training data.

Since it’s a two-stage training, the governing physics cannot be incorporated.

Outline

Physics-Informed Operator Learning on Latent Spaces

Part – I: Efficient algorithms beyond the existing ones

Part – II: Data-driven operator learning on reduced spaces

Part – III: Integrating physics and data to learn operator on reduced spaces

Our Proposed framework

Physics-Informed Latent Neural Operator: Integrating Physics and Data using Reduced Order Modeling

Manuscript in preparation

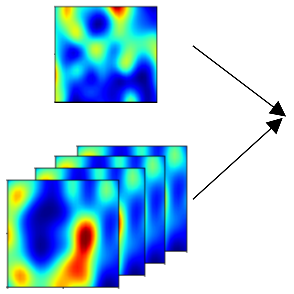
One-shot Learning: Physics Informed Latent Neural Operator

Operator $\mathcal{G} : \mathcal{X} \rightarrow \mathcal{Y}$

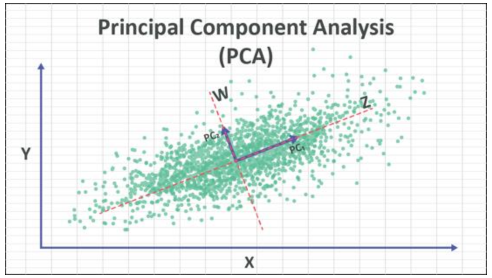
$\mathcal{G}_\theta : \mathcal{X} \rightarrow \mathcal{Y}, \quad \theta \in \Theta$

Training data $\{\mathbf{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$

N(way less than data-driven)

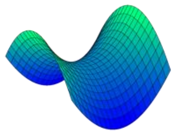


$\{\mathbf{y}_i\}_{i=1}^{N}$

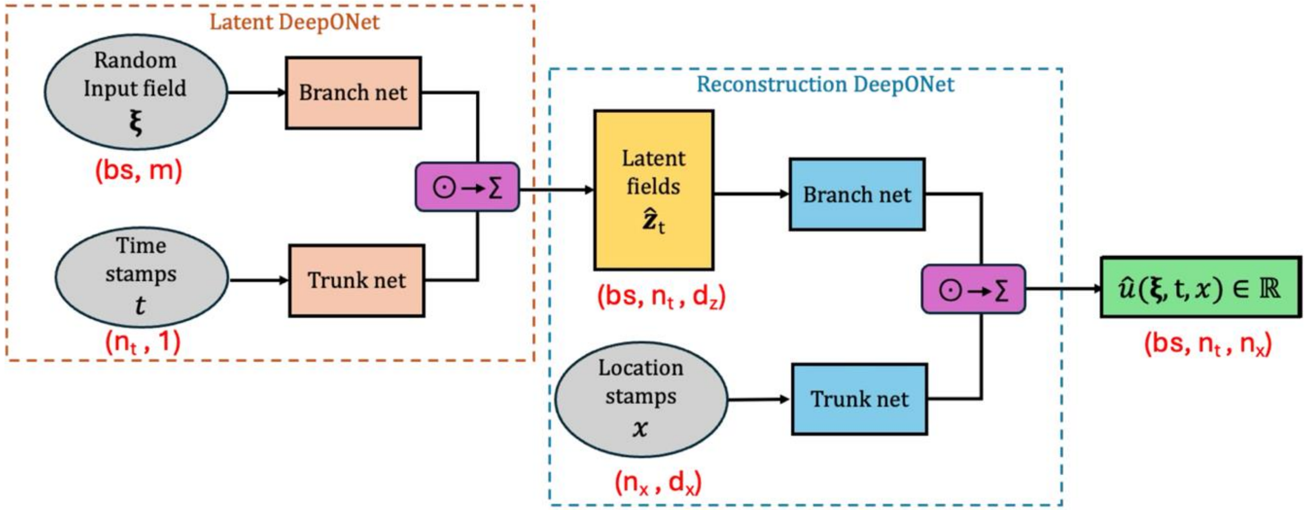


Or Autoencoder

Latent representation



$\{\mathbf{x}_i^r, \mathbf{y}_i^r\}_{i=1}^{N} \in \mathbb{R}^d$

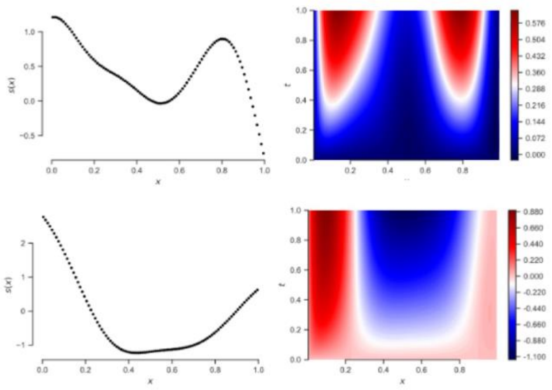
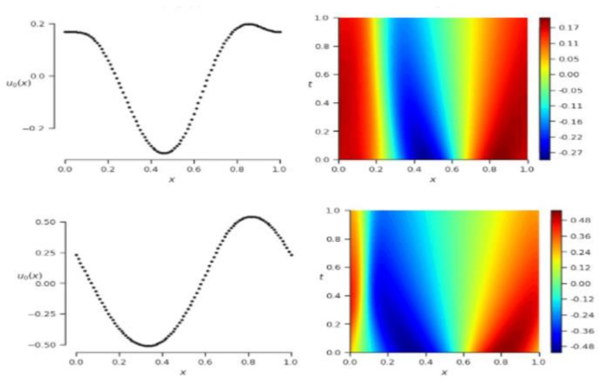
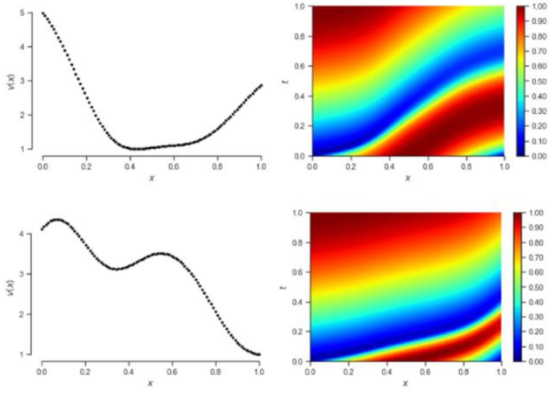


$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{data-driven}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{physics-informed}}(\boldsymbol{\theta}),$$

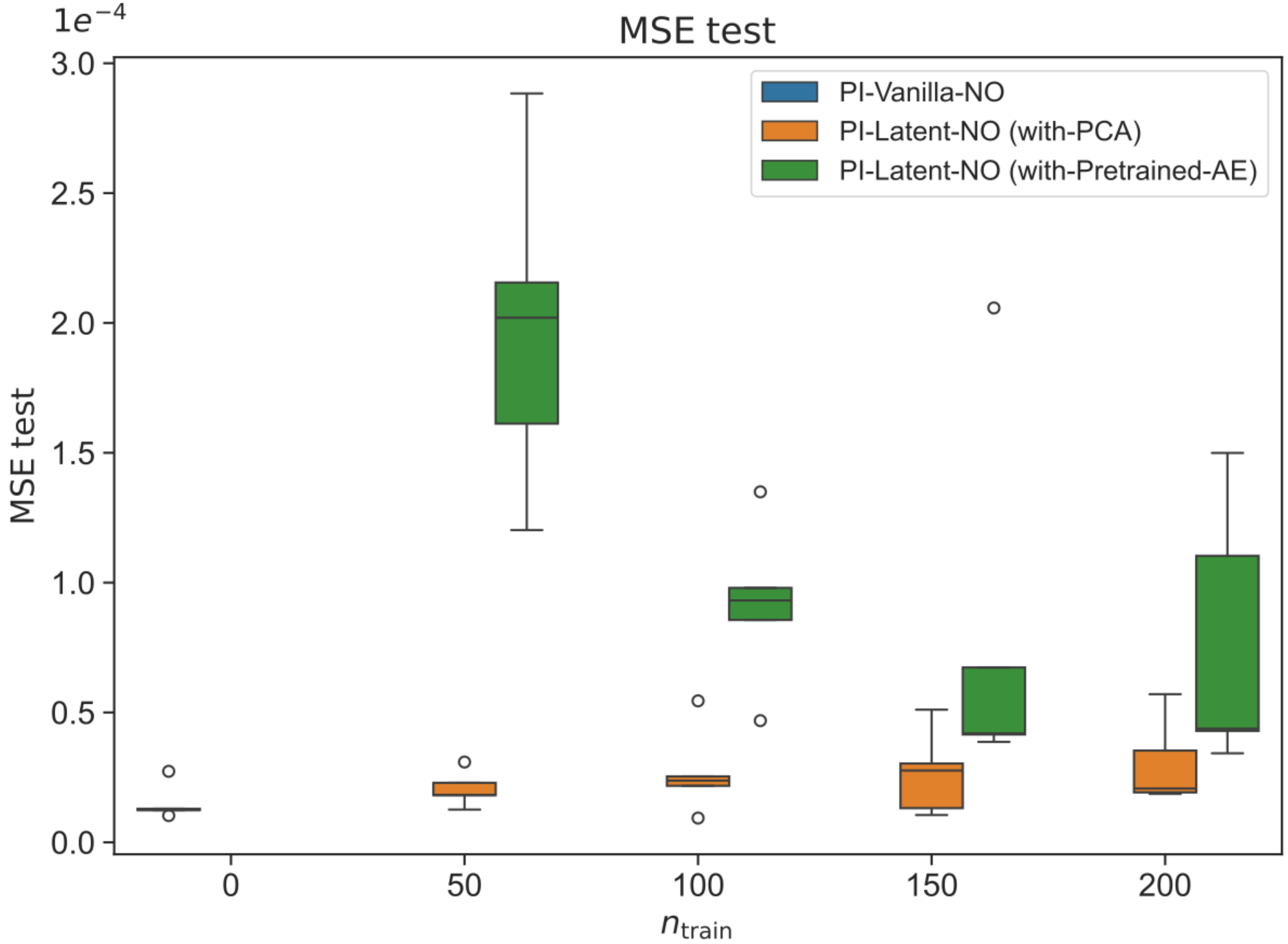
$$\mathcal{L}_{\text{data-driven}}(\boldsymbol{\theta}) = \frac{1}{n_{\text{train}}(n_t+1)} \sum_{i=1}^{n_{\text{train}}} \sum_{j=0}^{n_t} \left\| \mathbf{z}(\boldsymbol{\xi}^{(i)}, j\Delta t) - \hat{\mathbf{z}}(\boldsymbol{\xi}^{(i)}, j\Delta t) \right\|_2^2$$

$$+ \frac{1}{n_{\text{train}}(n_t+1)n_{\boldsymbol{x}}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=0}^{n_t} \sum_{k=1}^{n_{\boldsymbol{x}}} \left(u(\boldsymbol{\xi}^{(i)}, j\Delta t, \boldsymbol{x}^{(k)}) - \hat{u}(\boldsymbol{\xi}^{(i)}, j\Delta t, \boldsymbol{x}^{(k)}) \right)^2,$$

$$\mathcal{L}_{\text{physics-informed}}(\boldsymbol{\theta}) = \mathcal{L}_r(\boldsymbol{\theta}) + \mathcal{L}_{bc}(\boldsymbol{\theta}) + \mathcal{L}_{ic}(\boldsymbol{\theta}).$$

Case	Diffusion-reaction dynamics	Burgers' transport dynamics	Advection
PDE	$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + ku^2 + s(x),$ $D = 0.01,\ k = 0.01,$ $(t, x) \in (0, 1] \times (0, 1],$ $u(0, x) = 0,\ x \in (0, 1)$ $u(t, 0) = 0,\ t \in (0, 1)$ $u(t, 1) = 0,\ t \in (0, 1)$ $\mathcal{G}_{\boldsymbol{\theta}} : s(x) \to u(t, x).$	$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} - \nu\frac{\partial^2 u}{\partial x^2} = 0,$ $\nu = 0.01,$ $(t, x) \in (0, 1] \times (0, 1],$ $u(0, x) = g(x),\ x \in (0, 1)$ $u(t, 0) = u(t, 1)$ $\frac{\partial u}{\partial x}(t, 0) = \frac{\partial u}{\partial x}(t, 1)$ $\mathcal{G}_{\boldsymbol{\theta}} : g(x) \to u(t, x).$	$\frac{\partial u}{\partial t} + s(x)\frac{\partial u}{\partial x} = 0,$ $(t, x) \in (0, 1] \times (0, 1],$ $u(0, x) = \sin(\pi x)\ \forall\ x \in (0, 1),$ $u(t, 0) = \sin(0.5\pi t)\ \forall\ t \in (0, 1),$ $s(x) = v(x) - \min_x v(x) + 1$ $\mathcal{G}_{\boldsymbol{\theta}} : v(x) \to u(t, x).$
Input Function	$s(x) \sim \mathrm{GP}(0, k(x, x')),$ $\ell_x = 0.2,\ \sigma^2 = 1.0,$ $k(x, x') = \sigma^2 \exp\left\{-\frac{\Vert x - x'\Vert^2}{2\ell_x^2}\right\}.$	$g(x) \sim \mathcal{N}\left(0, 25^2\left(-\Delta + 5^2 I\right)^{-4}\right),$	$v(x) \sim \mathrm{GP}(0, k(x, x')),$ $\ell_x = 0.2,\ \sigma^2 = 1.0,$ $k(x, x') = \sigma^2 \exp\left\{-\frac{\Vert x - x'\Vert^2}{2\ell_x^2}\right\}.$
Samples			

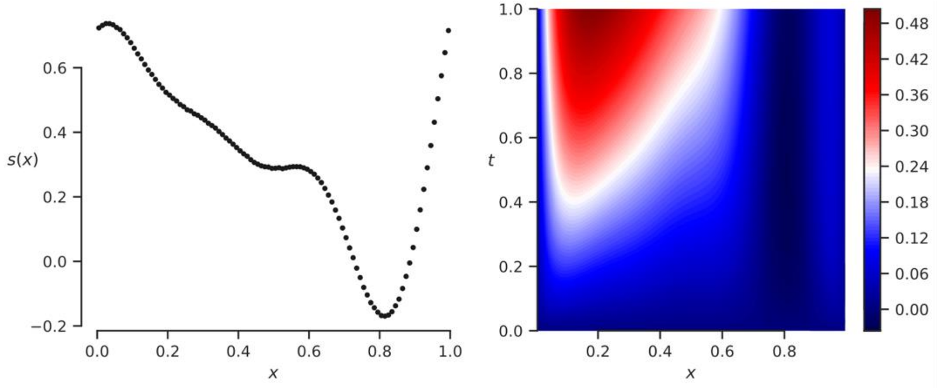
Accuracy Comparison



Reaction Diffusion Dynamics

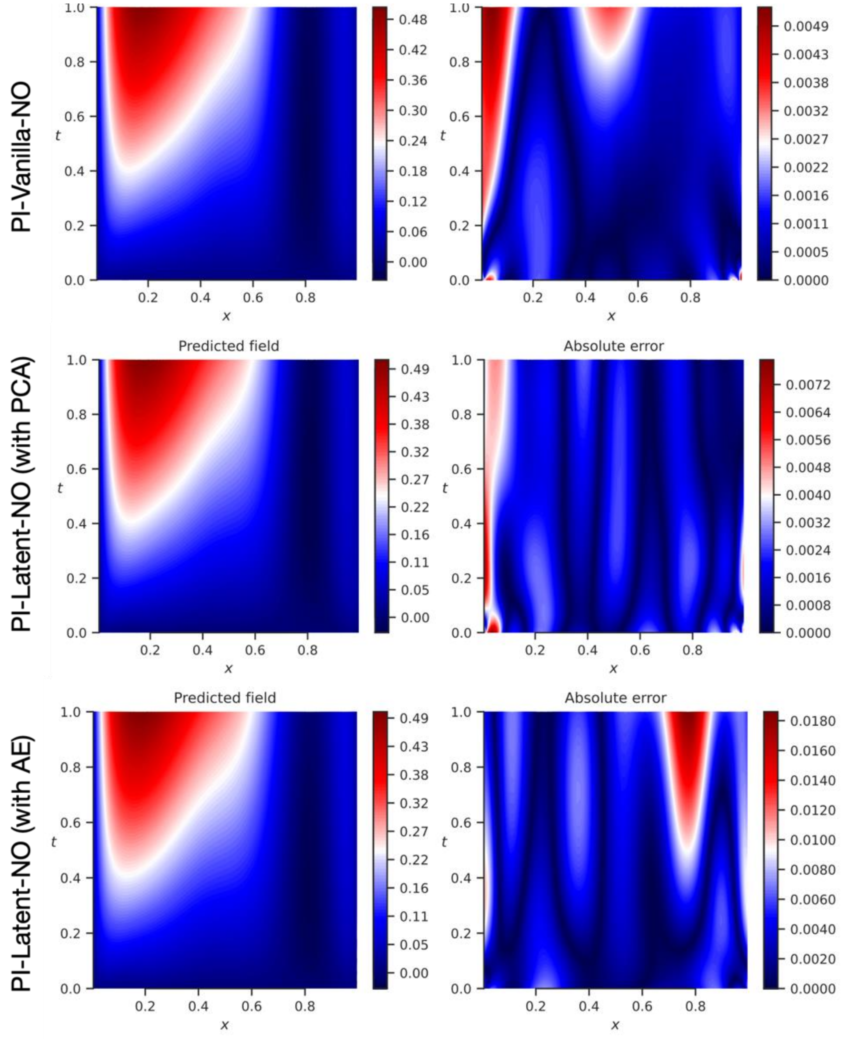
Source (Input function)

Solution (Ground Truth)

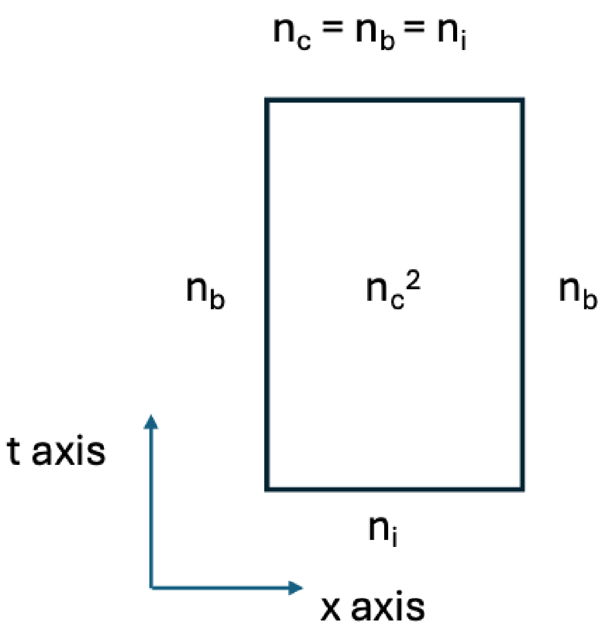


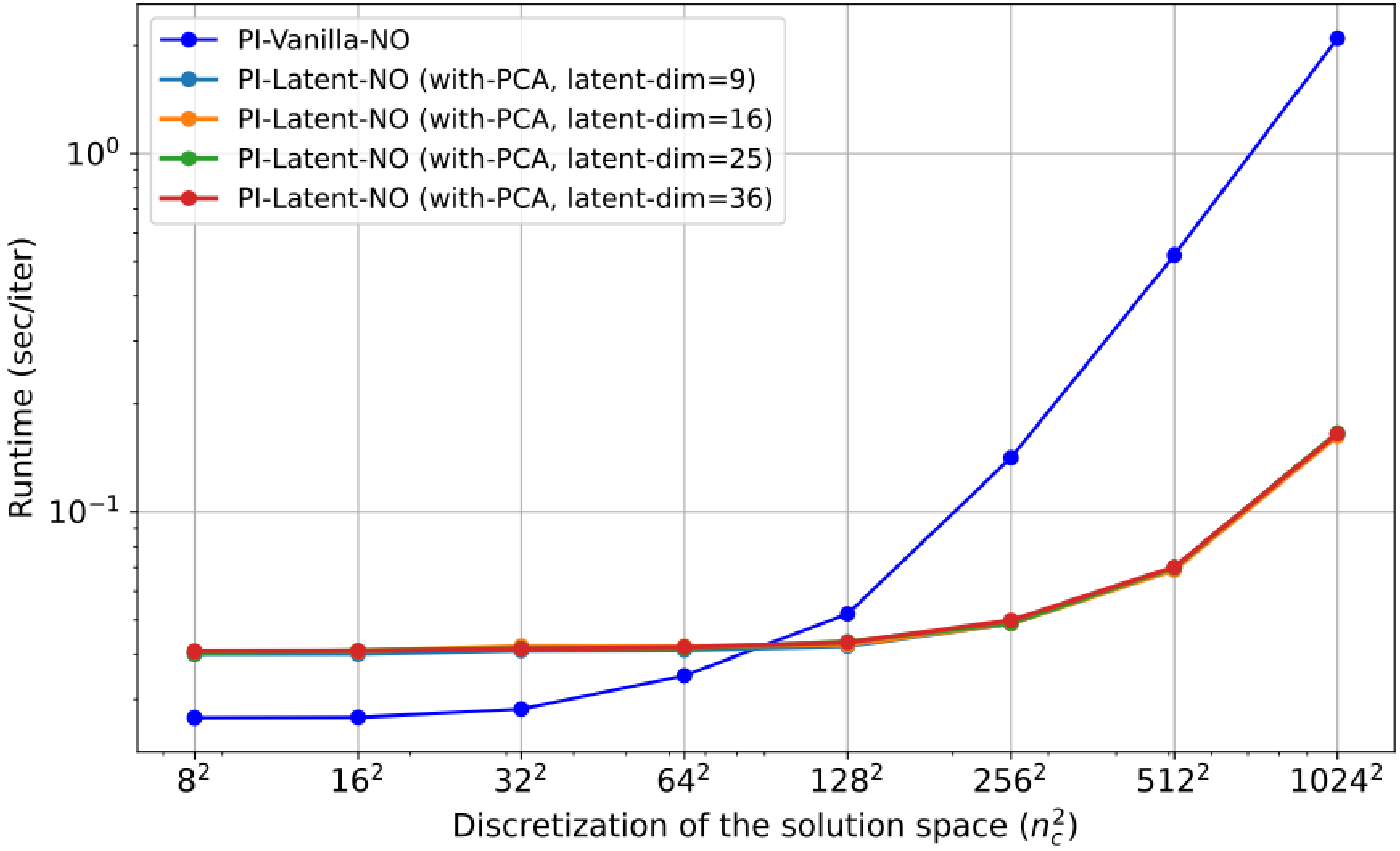
Prediction

Error

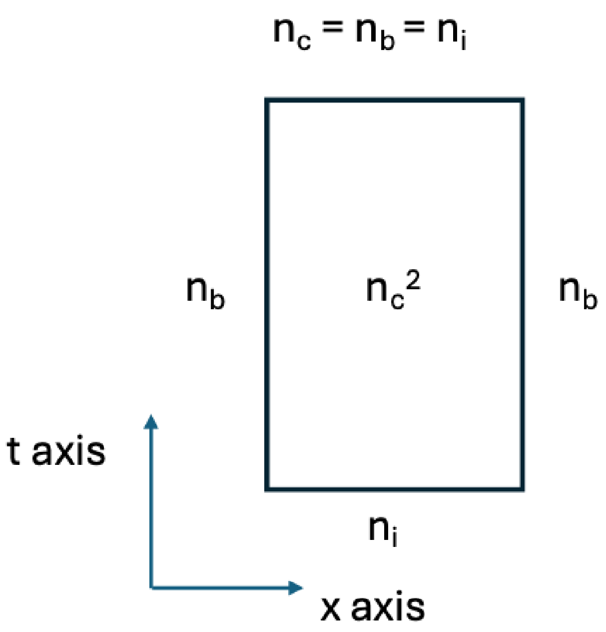


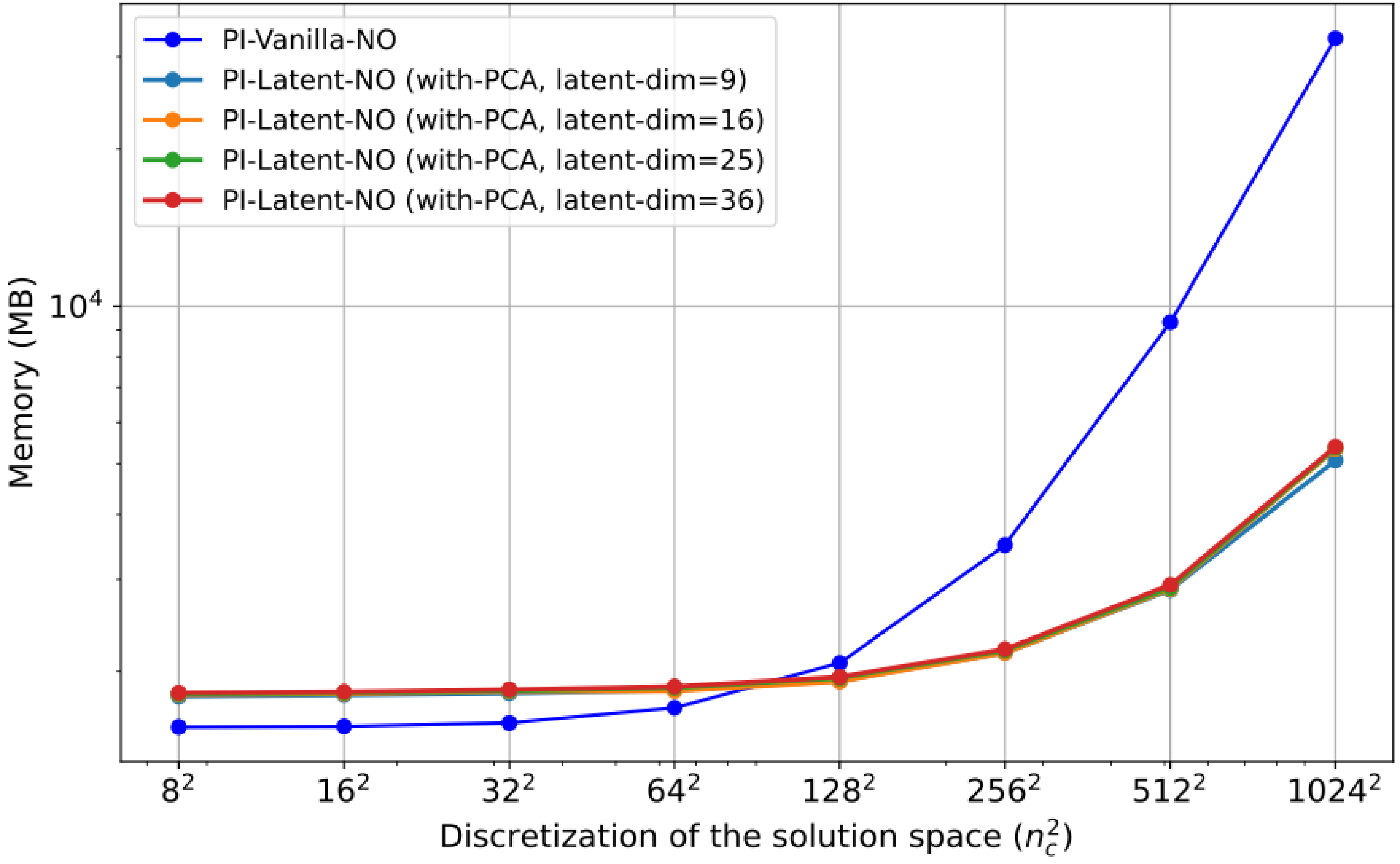
Runtime Scaling



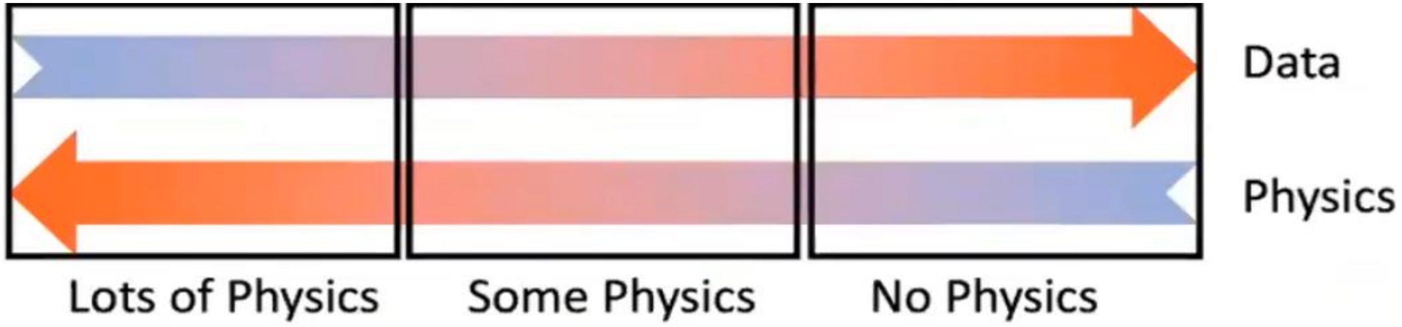


Memory Scaling





Key Takeaways



- These methods have a niche in real world problems, where partially physics in known and some measurements of quantities of interest are available.
- Separable architecture introduces the possibility of employing physics in neural operators.
- Learning NOs on reduced spaces with data and physics opens up the possibly of exploring large design spaces efficiently.

Acknowledgement








Funding

Thank you!