JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

July 15, 2025

Towards Foundation Models in Science: Bridging Neural Operators and Generalization

Somdatta Goswami
Civil and Systems Engineering
Johns Hopkins University

U.S. National Science Foundation

U.S. DEPARTMENT OF ENERGY

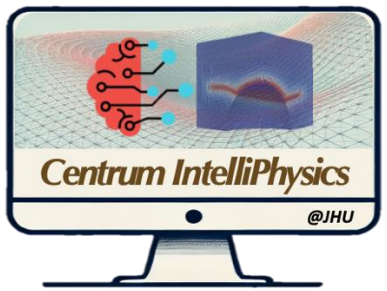
ICTP Advanced School on Foundation Models for Scientific Discovery

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







Physics-based Models

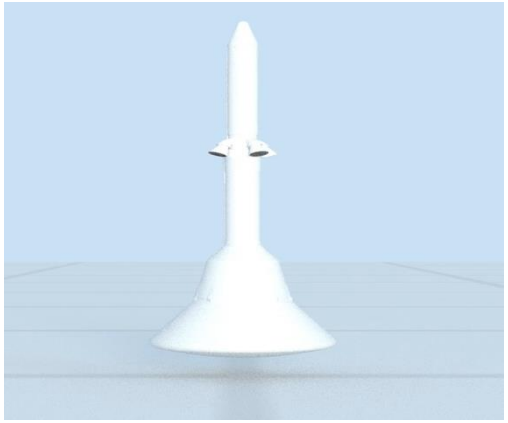
Can represent the Processes of Nature

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

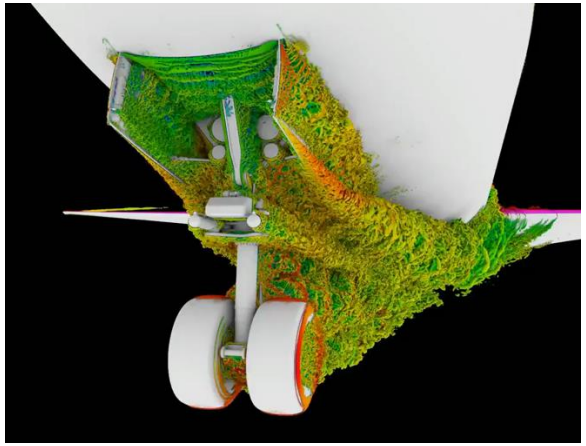
  To model earthquake: $m\frac{d^2u}{dt^2} + \mathrm{k}\frac{du}{dt} + F_0 = 0$

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

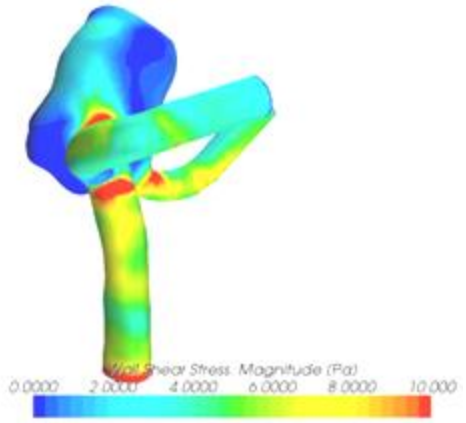
- Computational Mechanics helps us simulate these equations.



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

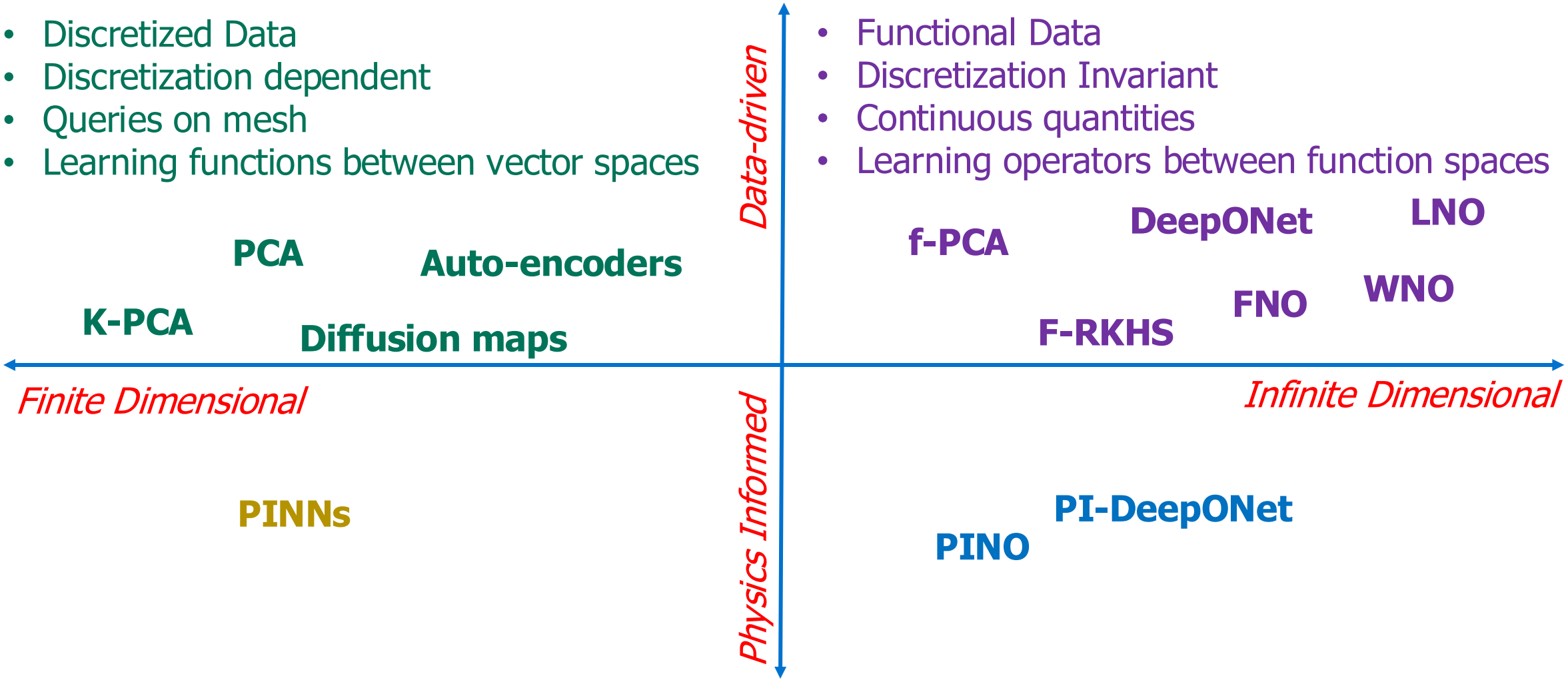
- Require knowledge of conservation laws, and boundary conditions
- Time consuming and strenuous simulations.
- Difficulties in mesh generation.
- Solving inverse problems or discovering missing physics can be prohibitively expensive.

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

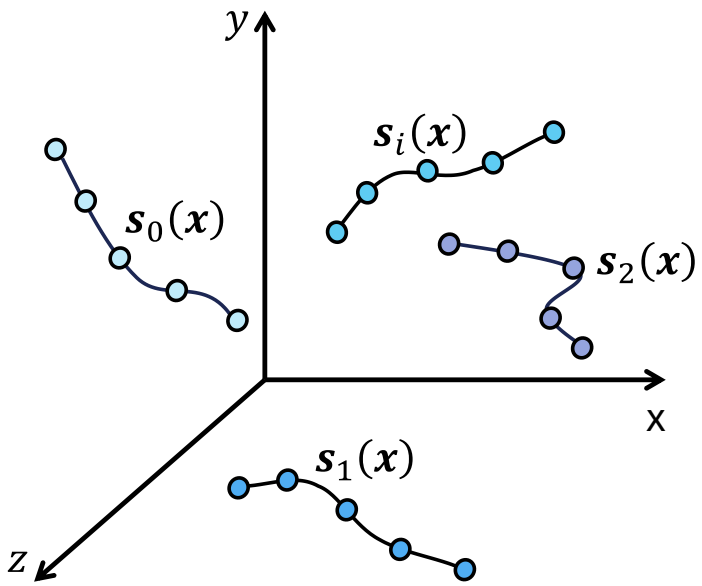
Surrogate Modeling Techniques

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

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



Operator Learning



Input space

Initial and Boundary conditions, Material parameters, etc.

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

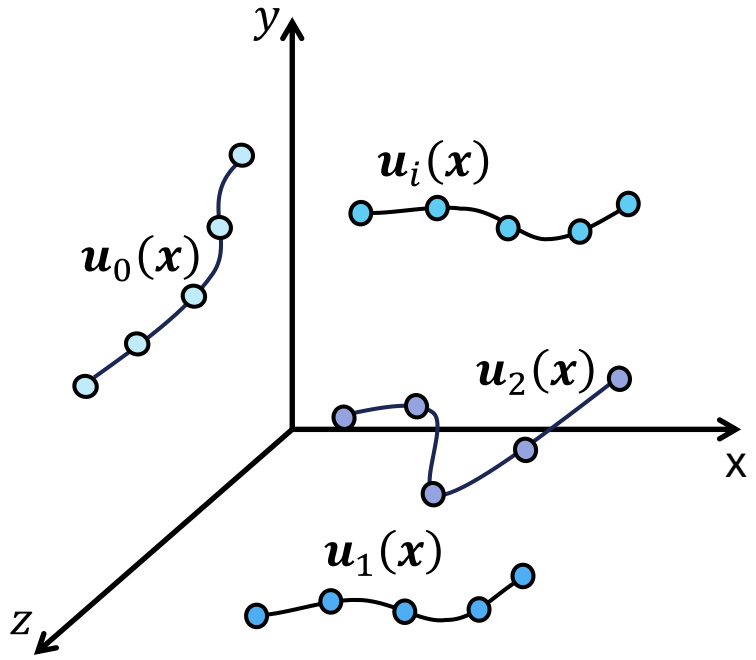
$$\Psi: \times\Theta \to \mathcal{U}$$

Training that

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

such

$$\Psi(., \theta^*) \approx \Phi$$



Output space

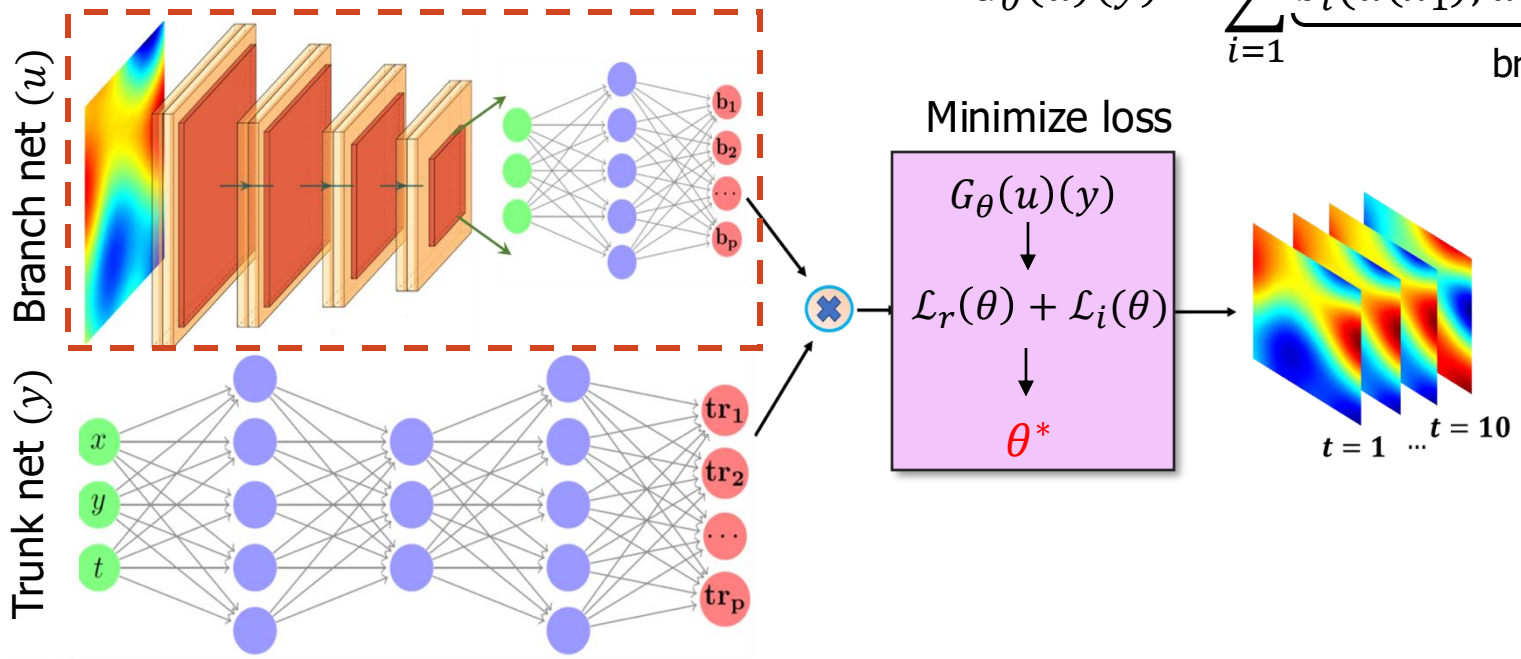
Displacement, Stress, Pressure, etc.

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 $\{y_i\}_{i=1}^m$

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

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



Data-Driven Training of DeepONet

$\nabla\big(K(\boldsymbol{x})\nabla u(\boldsymbol{x})\big) = 1\; u(\boldsymbol{x}) = 0\;\forall\; \boldsymbol{x} \in \partial\Omega$

Nonlinear operator $\mathcal{G} : \mathcal{K} \to \mathcal{U}$

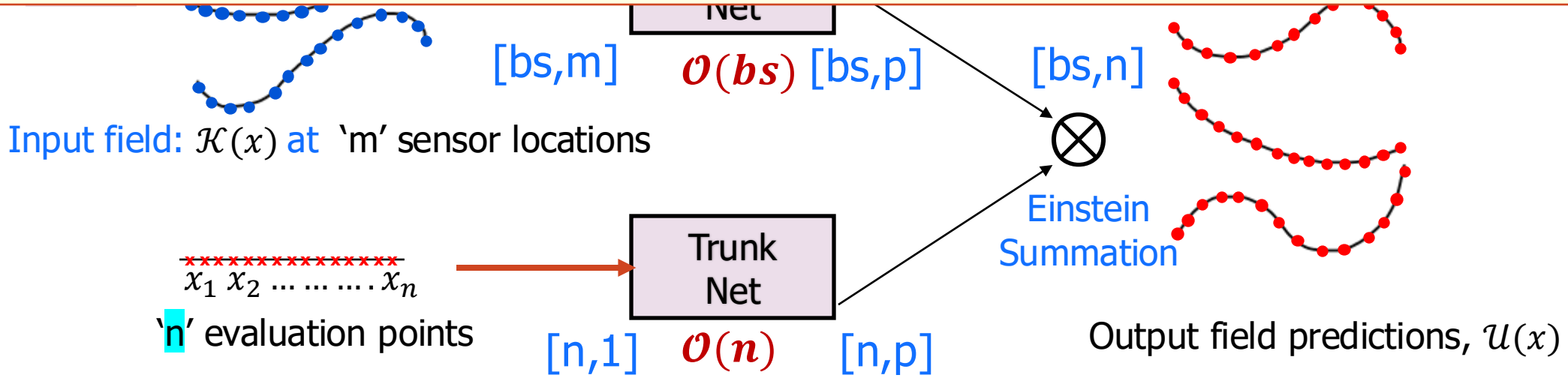
Neural operator $\mathcal{G}_\theta : \mathcal{K} \to \mathcal{U},\quad \theta \in \Theta$

Training data $\{K_i(x_j), U_i(x_j)\}_{i=1,j=1}^{N,n}$

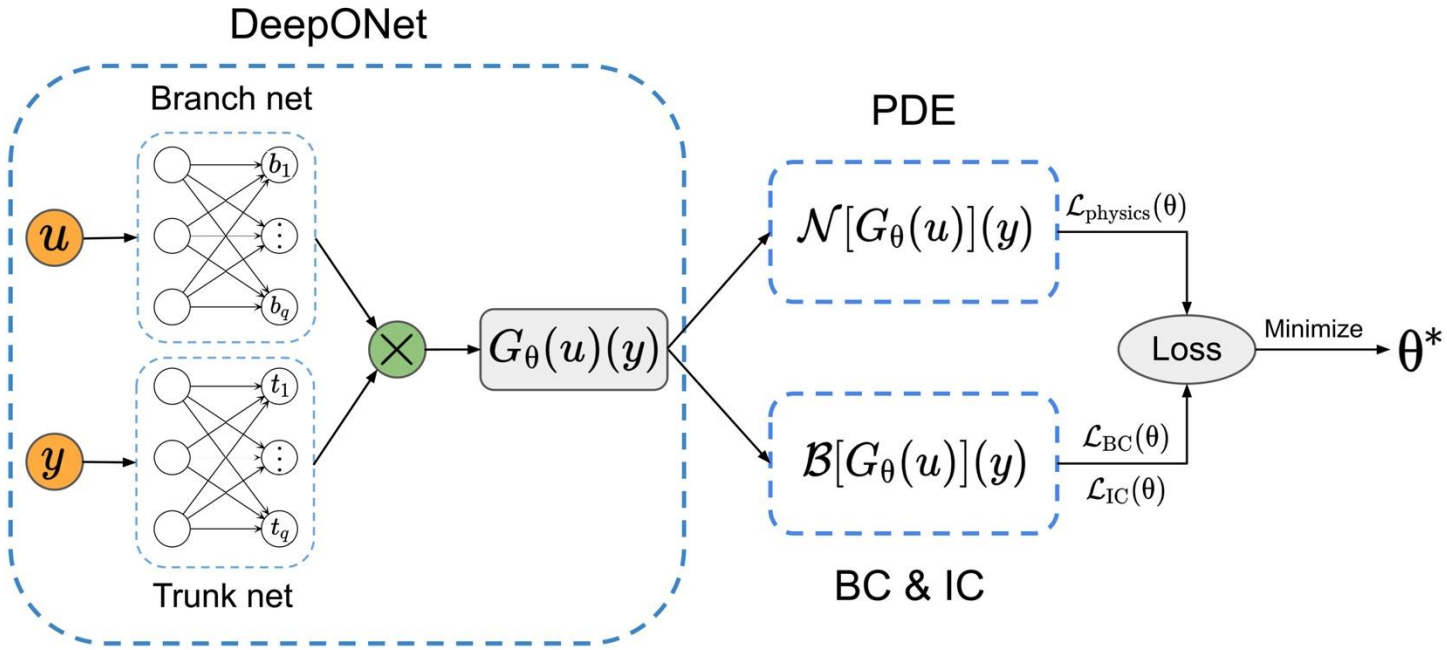
Training Dataset

S.No	Input field data	Output field data
1	$k^1(x_1), \mathrm{k}^1(x_2), \cdots\cdots\cdots, k^1(x_m)$	$\mathrm{u}^1(x_1), \mathrm{u}^1(x_2), \cdots\cdots\cdots, \mathrm{u}^1(x_n)$
2	$k^2(x_1), \mathrm{k}^2(x_2), \cdots\cdots\cdots, \mathrm{k}^2(x_m)$	$u^2(x_1), \mathrm{u}^2(x_2), \cdots\cdots\cdots, \mathrm{u}^2(x_n)$
.	.	.
N	$k^{\mathrm{N}}(x_1), \mathrm{k}^{\mathrm{N}}(x_2), \cdots\cdots\cdots, \mathrm{k}^{\mathrm{N}}(x_m)$	$u^N(x_1), u^N(x_2), \cdots\cdots\cdots, \mathrm{u}^{\mathrm{N}}(x_n)$

Extremely data-hungry.

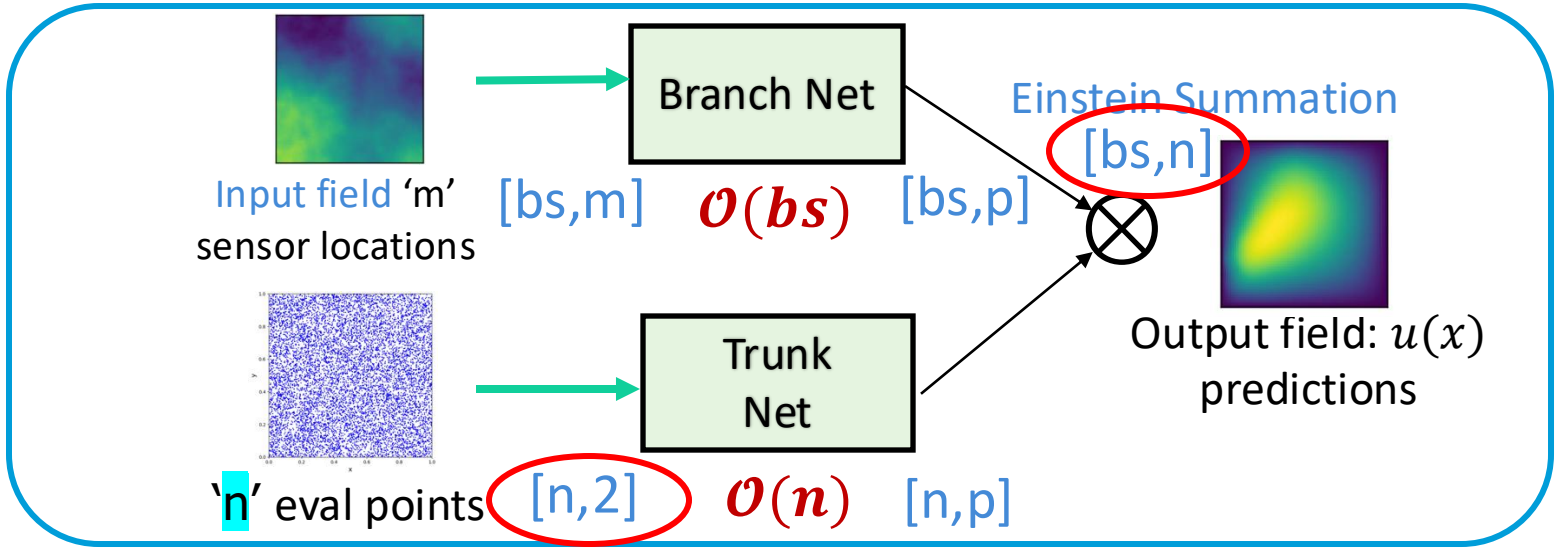


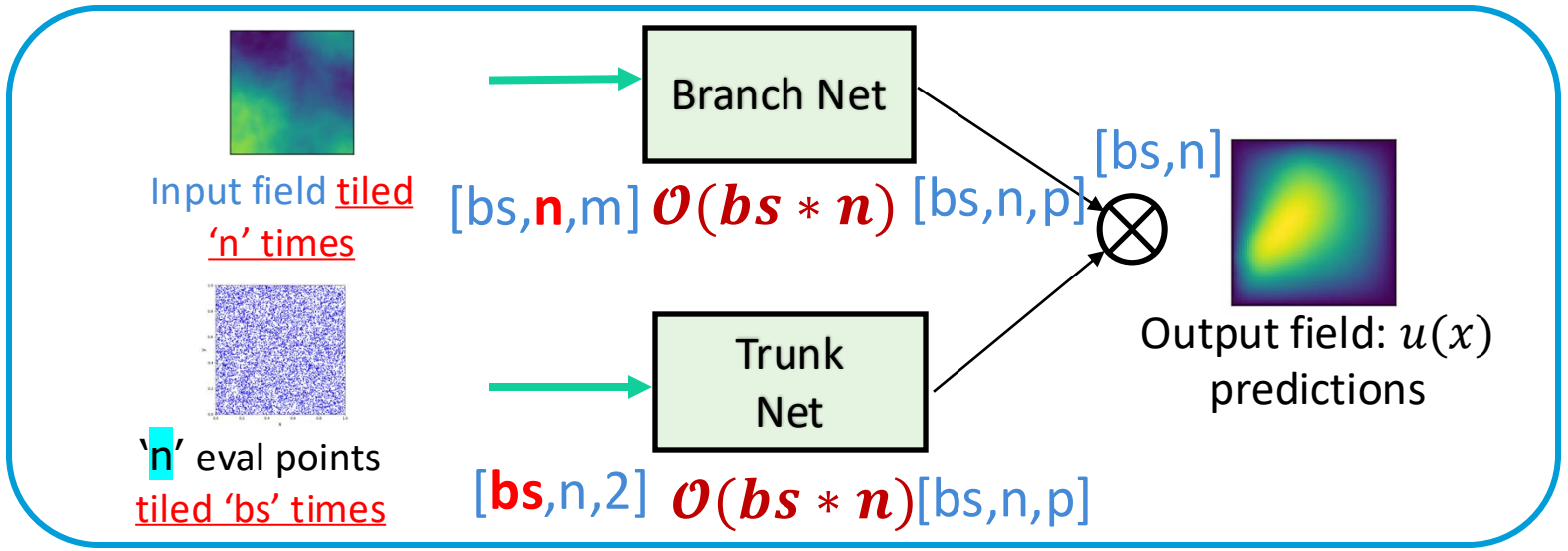
Physics-Informed DeepONet



- 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.

Frameworks for $\nabla\big(K(\boldsymbol{x})\nabla u(\boldsymbol{x})\big) = 1\ u(\boldsymbol{x}) = 0\ \forall\ \boldsymbol{x} \in \partial\Omega$ and $\boldsymbol{x} = (x, y)$





Derivatives

$$\frac{\partial \mathcal{L}}{\partial \theta}$$

$$\frac{\partial \mathcal{L}}{\partial \theta}, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \dots$$

Reverse-mode autodiff

$$J = [bs * n, bs * n]$$

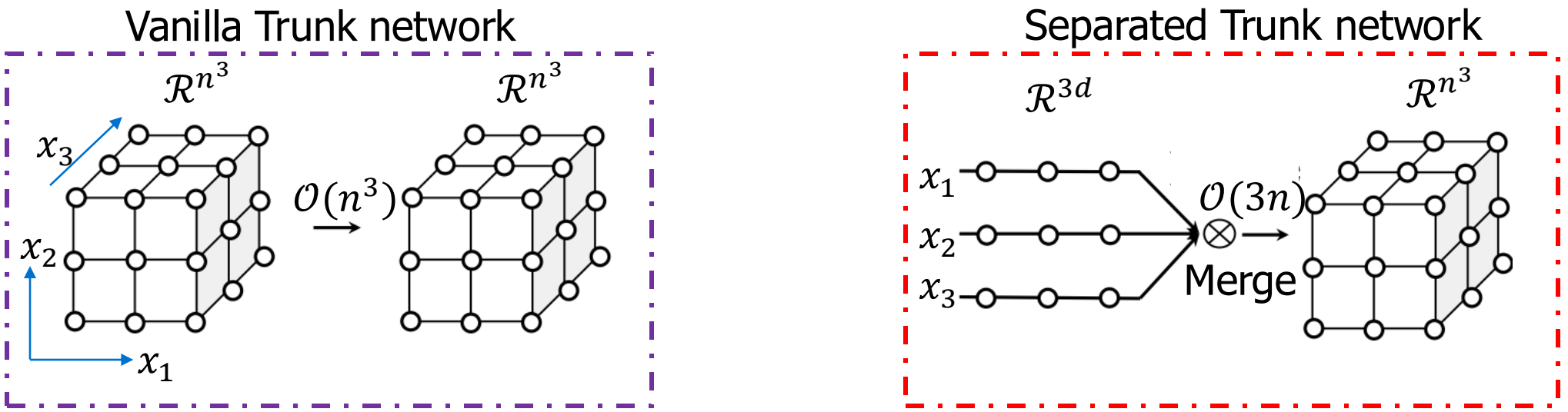
Our Proposed framework

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

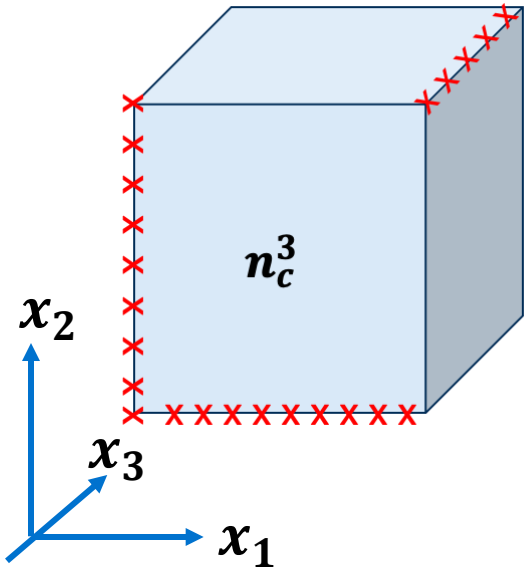
Luis Mandl [a], Somdatta Goswami [b], Lena Lambers [a], Tim Ricken [a]

Introducing Separation of Variables



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.

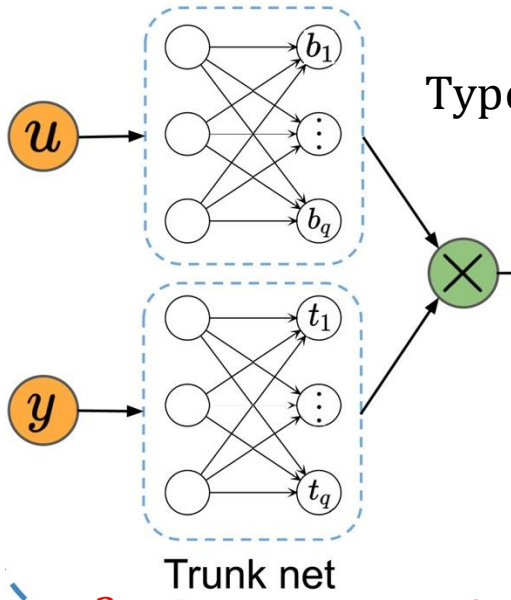
Vanilla – Physics Informed DeepONet



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

$\mathcal{O}(bs * n_c^3)$

[bs,n_c^3,m] Branch net [bs,n_c^3,p]



Trunk net

[bs,n_c^3,3] [bs,n_c^3,p]

$\mathcal{O}(bs * n_c^3)$

Type equation here.

$G_\theta(u)(y)$

[bs, n_c^3]

Jacobian size: [bs $* n_c^3$, bs $* n_c^3$]

PDE

$\mathcal{N}[G_\theta(u)](y)$

$\mathcal{L}_{\text{physics}}(\theta)$

$\mathcal{B}[G_\theta(u)](y)$

$\mathcal{L}_{\text{BC}}(\theta)$

$\mathcal{L}_{\text{IC}}(\theta)$

BC & IC

Loss

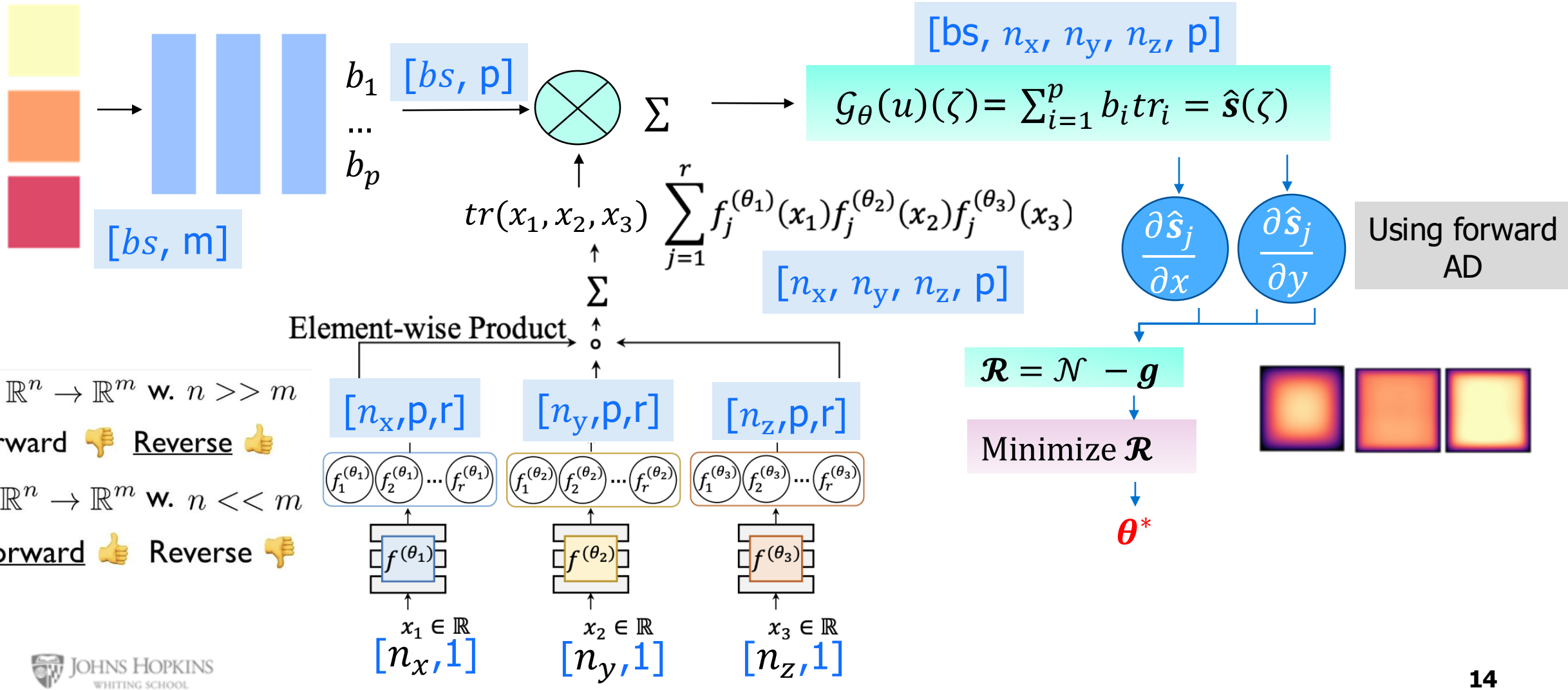
Minimize

θ^*

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.

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	2	5.1e-2	136.6
	Separable (Ours)		6.2e-2	3.64
Consolidation Biot's Theory	Vanilla	2	7.7e-2	169.43
	Separable (Ours)		7.9e-2	3.68
Parameterized Heat Equation	Vanilla	4	-	10,416.7
	Separable (Ours)		7.7e-2	91.73

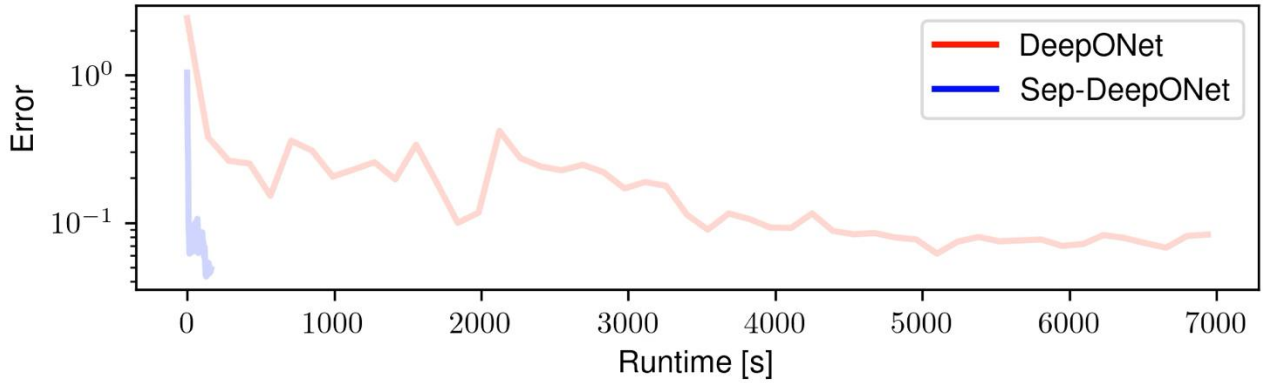
Burgers' Equation

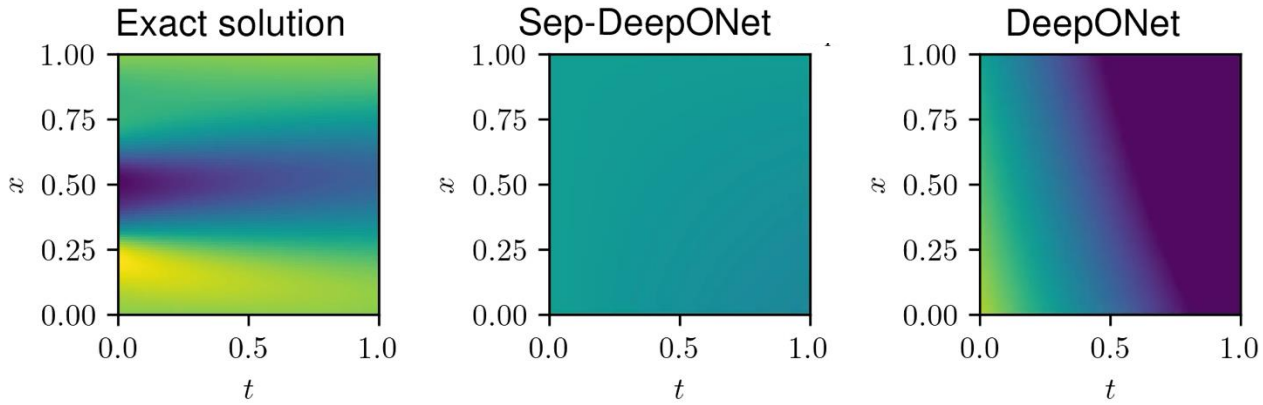
$$\frac{\partial s(x,t)}{\partial t} + s\frac{\partial s(x,t)}{\partial x} - \nu\frac{\partial^2 s(x,t)}{\partial x^2} = 0,$$

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

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

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

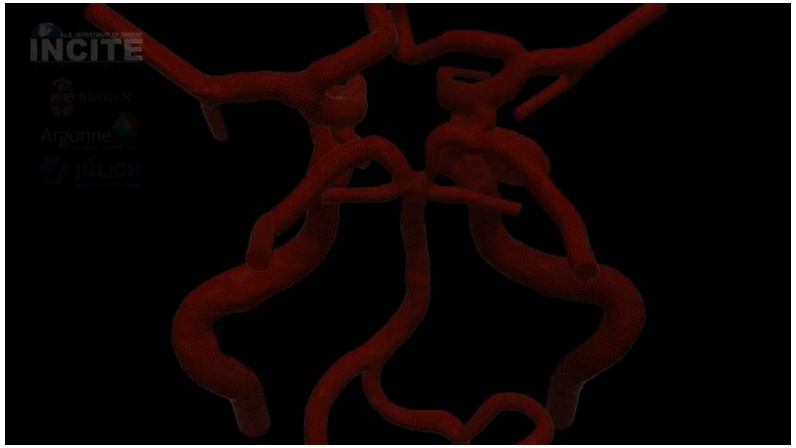


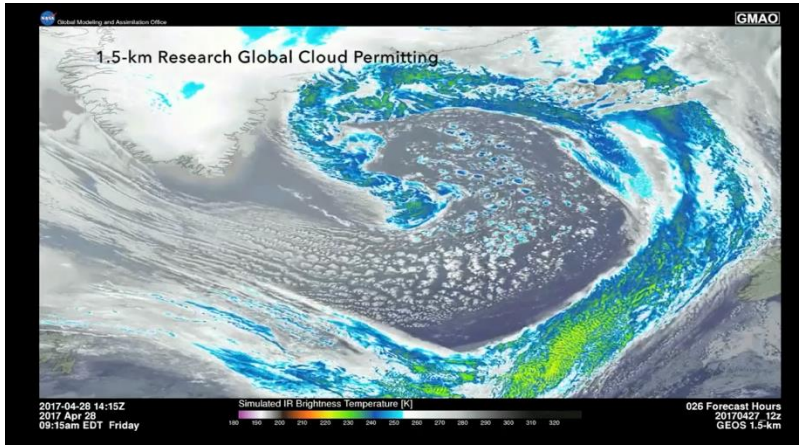


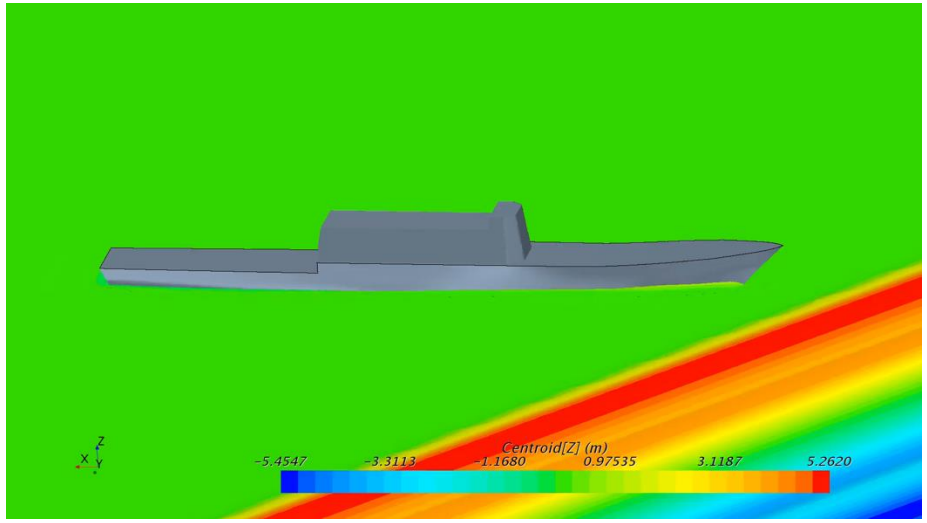
Model	Branch	Trunk	p	r	Parameters	$\mathcal{L}_2$ rel. err.	Runtime [s]	Runtime improvment
Vanilla PI-DeepONet	6×[100]	6×[100]	100	-	131,701	5.14e-2	6,829.2	-
Sep-PI-DeepONet	6×[100]	6×[100]	50	50	672,151	6.24e-2	182.1	97,33%
	6×[100]	6×[100]	20	20	244,921	6.04e-2	197.8	97,10%
	6×[100]	6×[50]	20	20	129,221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

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

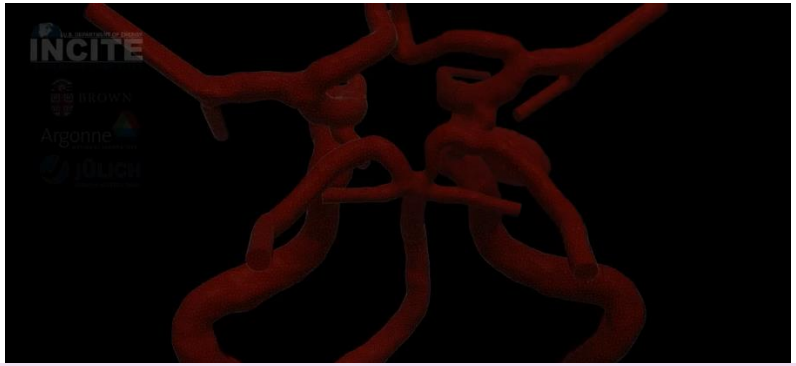
Can we harness the explosion of data to extract knowledge, insight and decision?

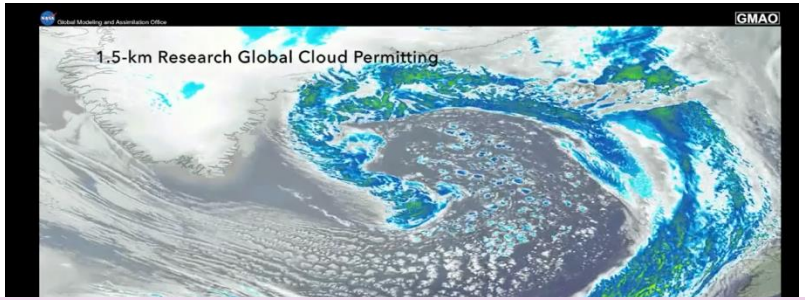

INCITE


Global Modeling and Assimilation Office
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


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

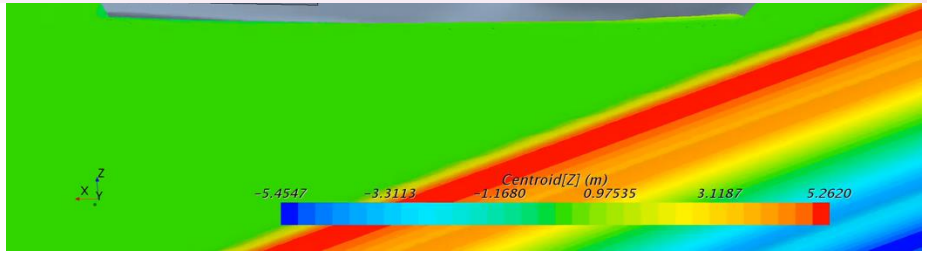

StreamTime LIVE - Baltimore, MD 2024-03-26 01:28:45 EDT





BIG Decisions need BIG MODELS

But we have: sparse high-dimensional datasets



Our Proposed framework

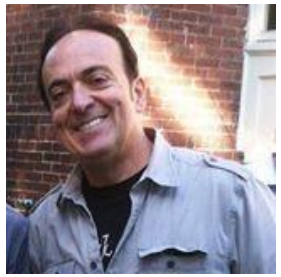
nature communications

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

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

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

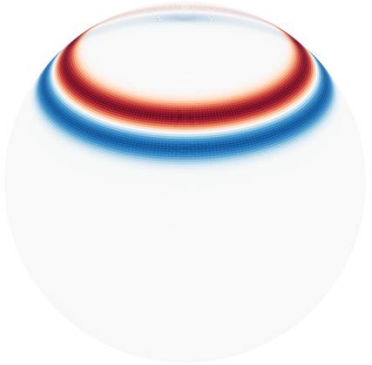
$$h'(\lambda,\phi) = \hat{h}\cos(\phi)\mathrm{e}^{-(\lambda/\alpha)^2}\mathrm{e}^{-[(\phi_2-\phi)/\beta]^2}$$

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

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

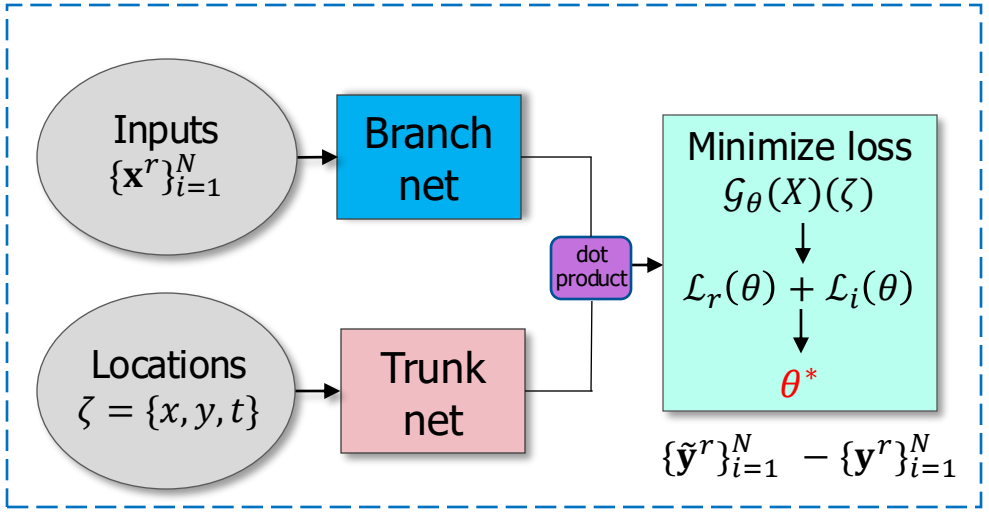
Atmospheric Flow

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

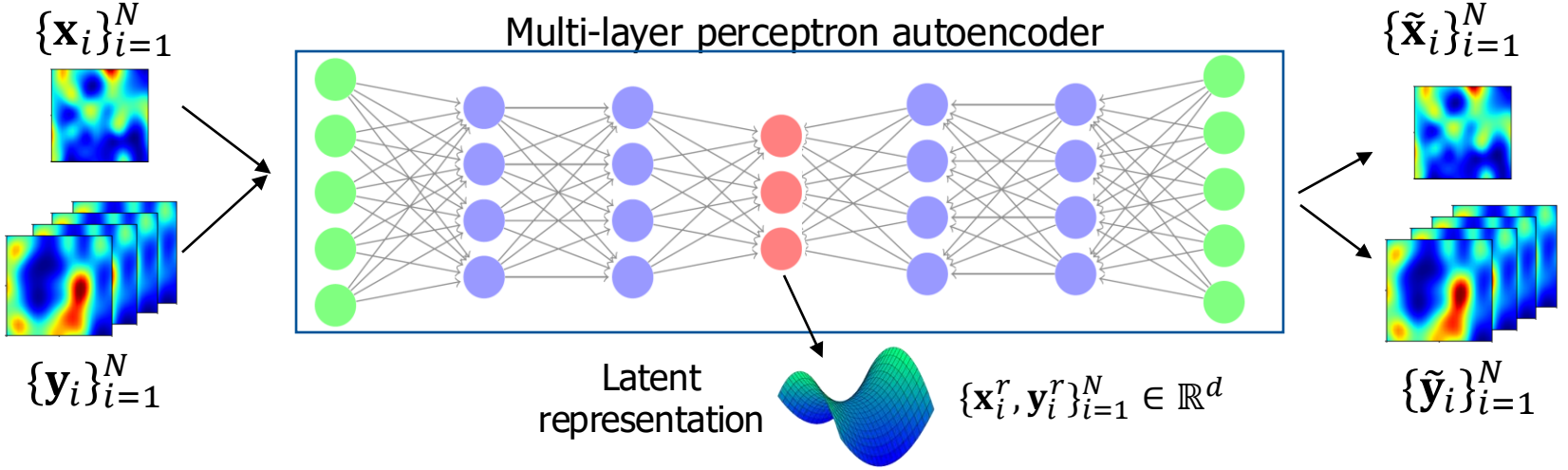


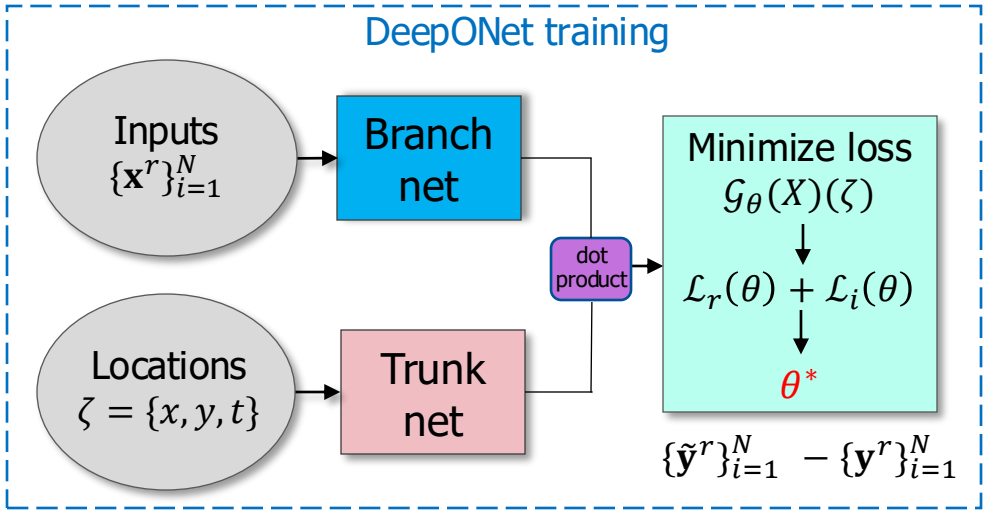
Latent DeepONet for time-dependent PDEs

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

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

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



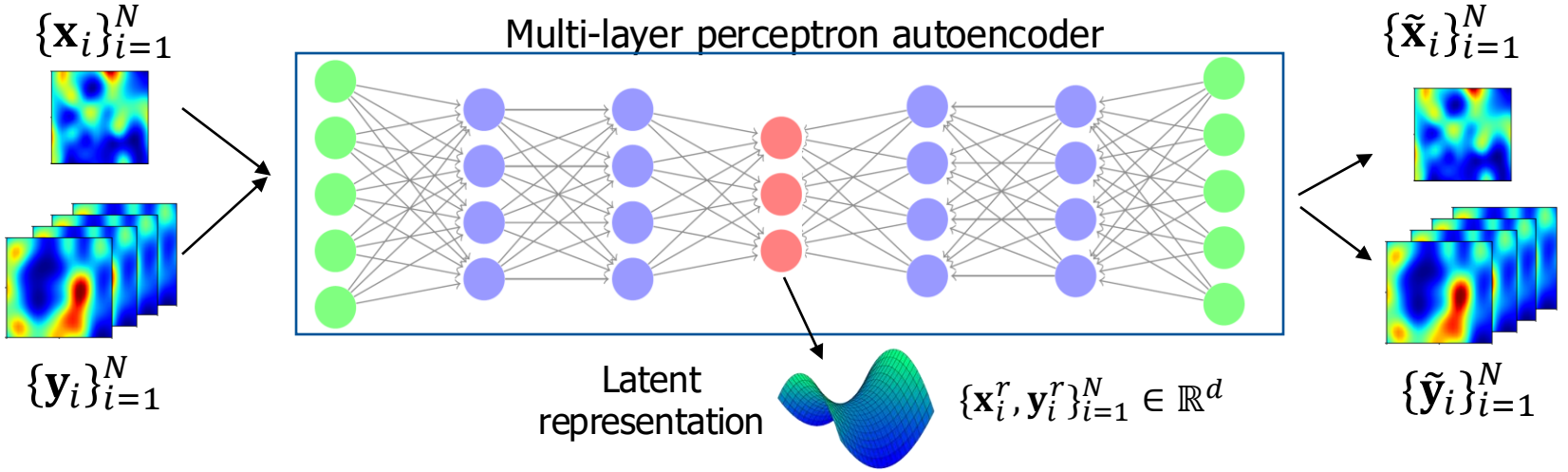


Latent DeepONet for time-dependent PDEs

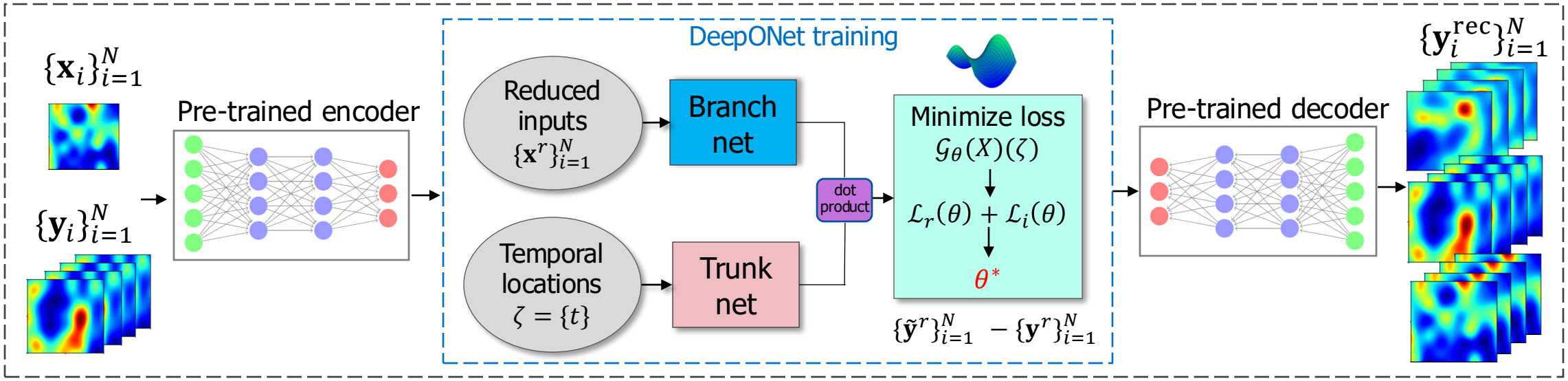
Operator $\mathcal{G}: \mathcal{X} \to \mathcal{Y}$

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

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



Latent DeepONet



MLAE: multi-layer autoencoders
PCA: principal component analysis

Consolidated results

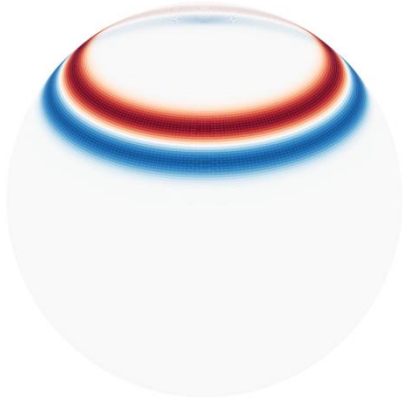
Accuracy of *L-DeepONet for MLAE and PCA*

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

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

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

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

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

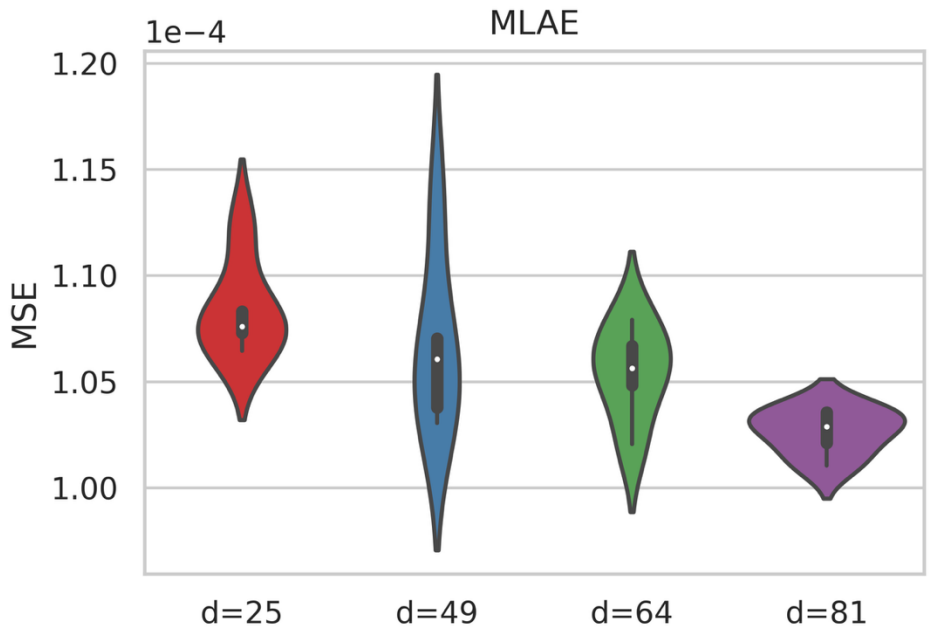
Results

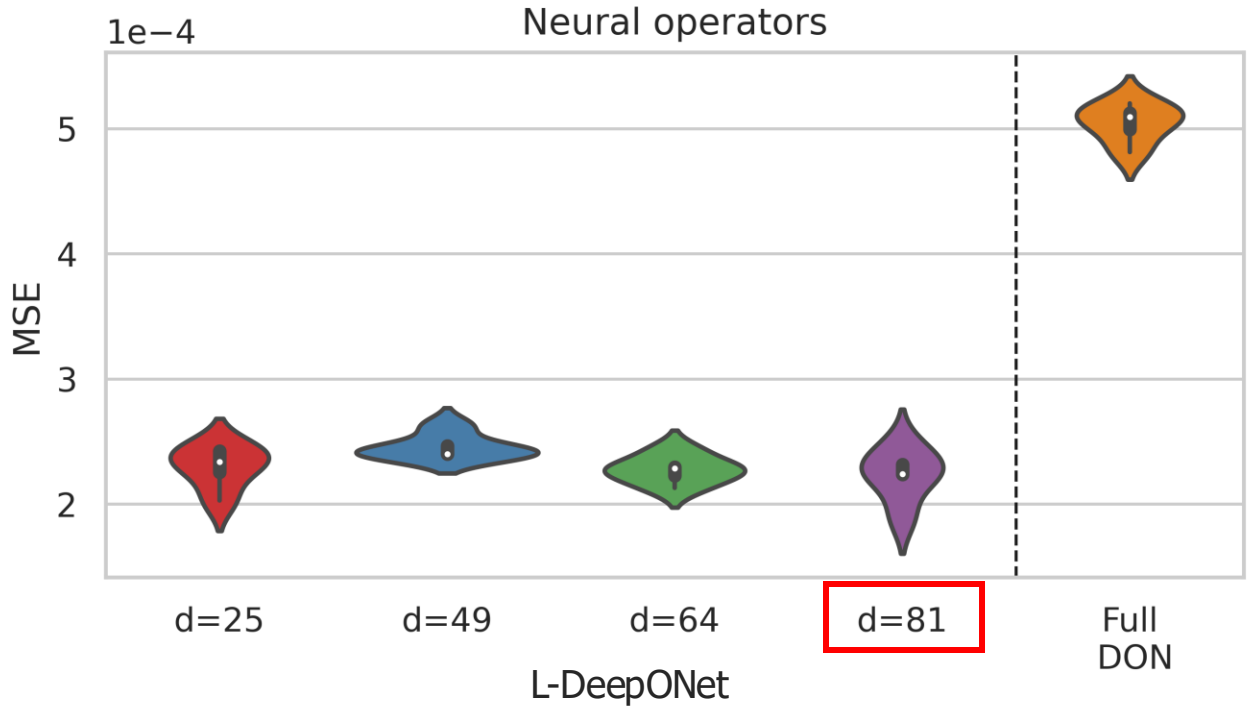
- $\Omega = [0,2\pi]x[0,2\pi]$, $(n_x \times n_y) = (256x256)$ mesh points
- Output dimensionality: $72x256x256 = 4{,}718{,}592$
- Simulation: $t = [0,360h], \delta t = 0.1\bar{6}h$, Time steps: $n_t = 72$

Training Time (seconds)

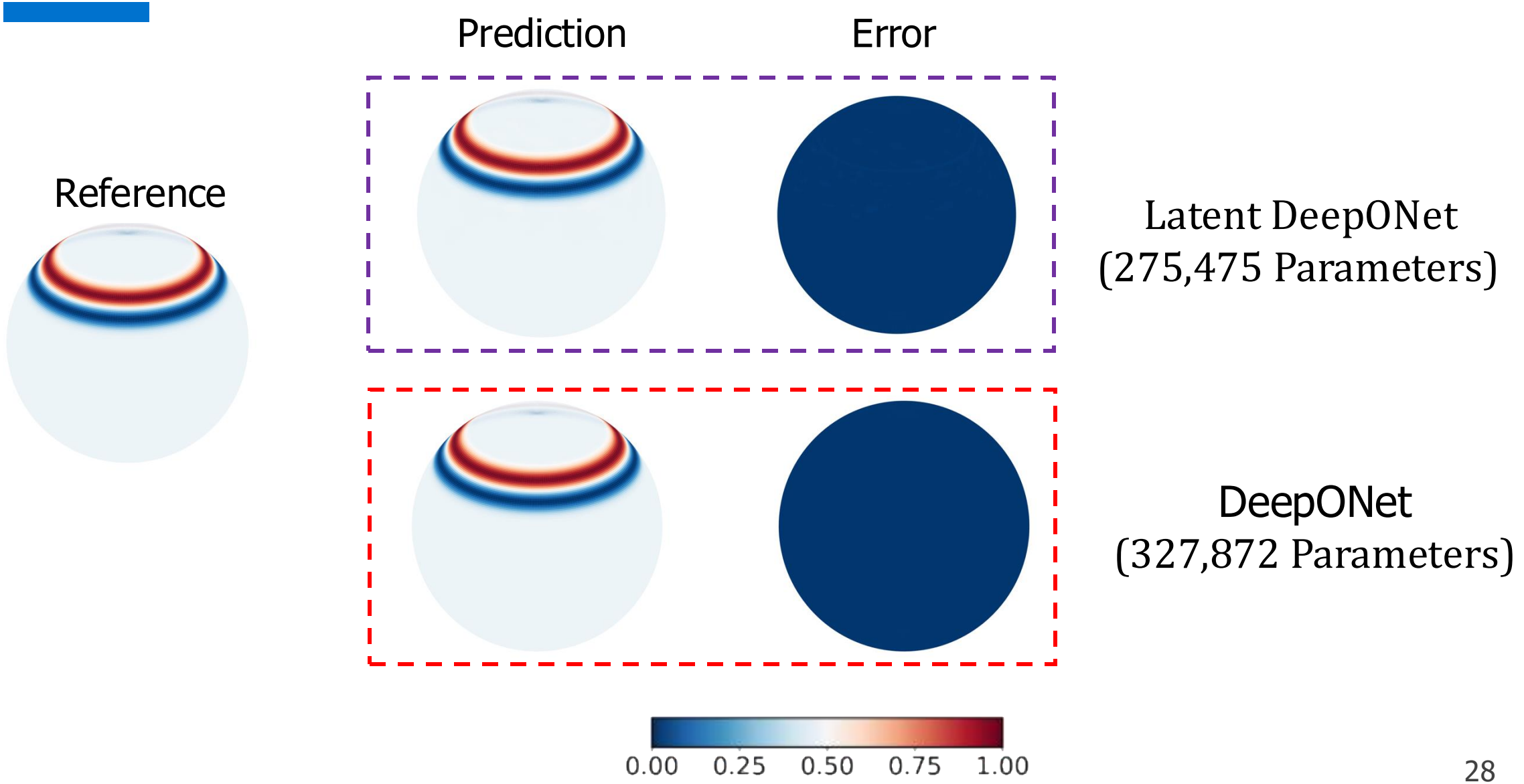
MLAE + Latent DON: 15, 218

Full DON: 379,022

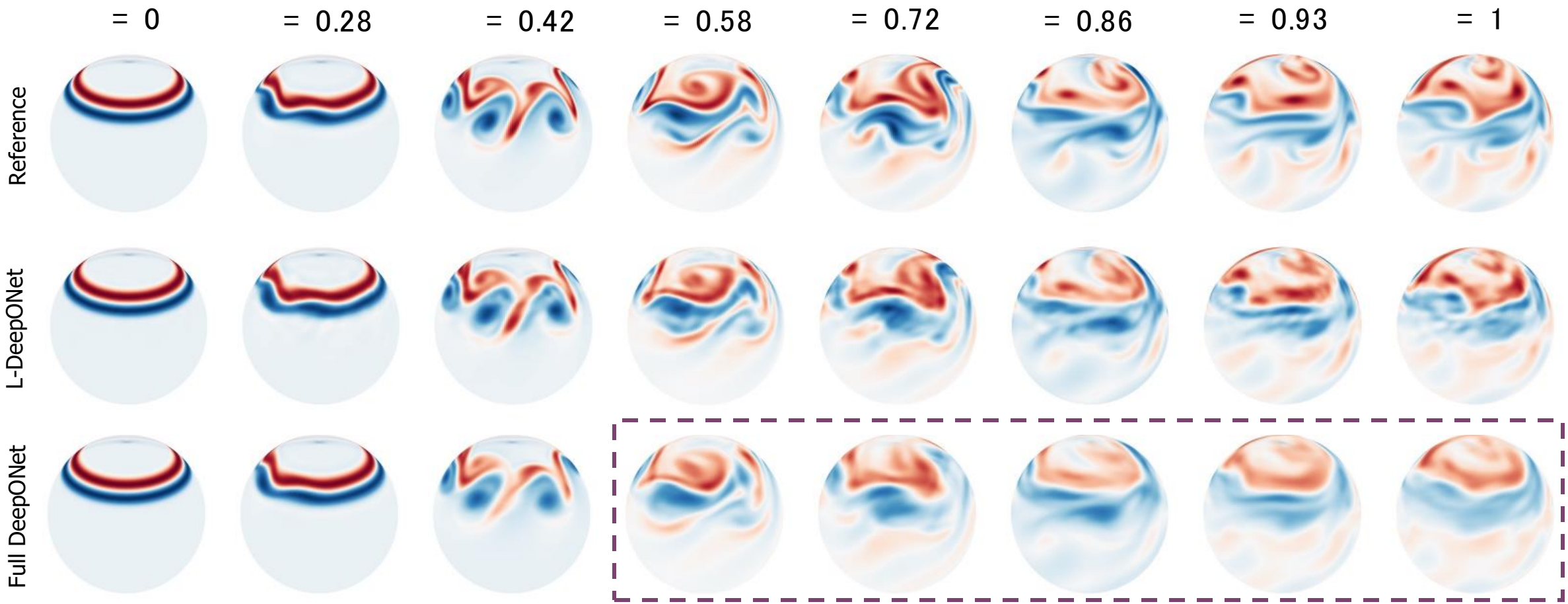




Results


Prediction
Error
Reference
Latent DeepONet
(275,475 Parameters)
DeepONet
(327,872 Parameters)
0.00
0.25
0.50
0.75
1.00

Latent DeepONet and Full DeepONet



Shortcomings

The framework requires voluminous training data.

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

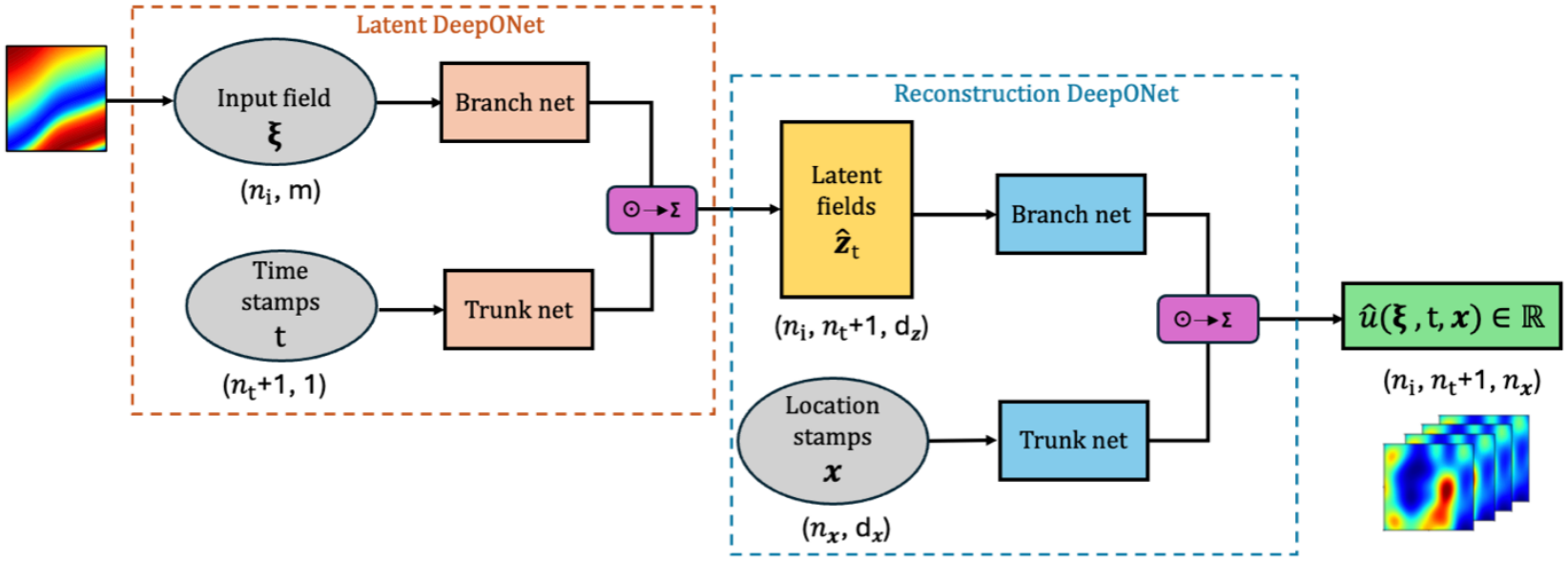
Our Proposed framework

Physics-Informed Latent Neural Operator for Real-time Predictions of Complex Physical Systems

Sharmila Karumuri[a], Lori Graham-Brady[a], Somdatta Goswami[a,*]

[a] *Johns Hopkins University, Department of Civil and Systems Engineering, Baltimore, 21218, Maryland, USA*

Physics-Informed Latent Neural Operator



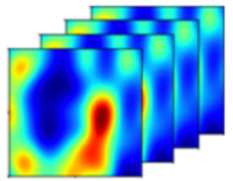
Schematic of Our Approach

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

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

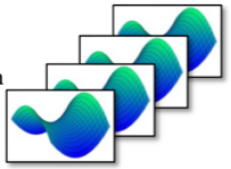
$$\mathcal{L}_{\text{data-driven}}(\boldsymbol{\theta}) = \frac{1}{n_{\text{train}}(n_t+1)n_{\boldsymbol{x}}} \sum_{i=1}^{n_{\text{train}}} \sum_{j=0}^{n_t} \sum_{k=1}^{n_{\boldsymbol{x}}} \left(u(\boldsymbol{\xi}^{(i)}, j\Delta t, \boldsymbol{x}^{(k)}) - \hat{u}(\boldsymbol{\xi}^{(i)}, j\Delta t, \boldsymbol{x}^{(k)})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

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



Dimensionality reduction

PCA/Autoencoder

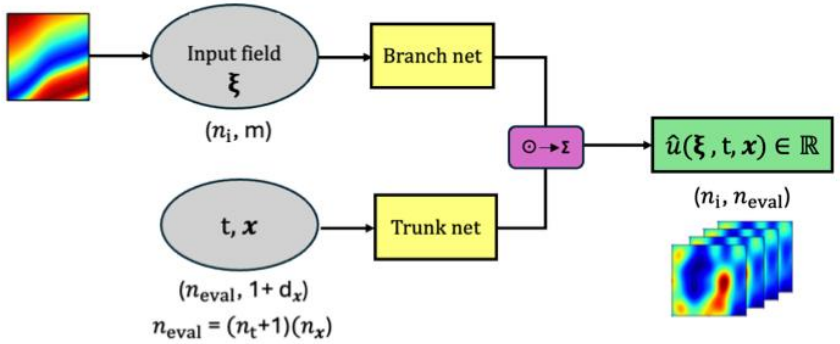


$$\mathcal{T}_{u,i} = \left[\mathbf{u}_0^{(i)}, \mathbf{u}_{1\Delta t}^{(i)}, \mathbf{u}_{2\Delta t}^{(i)}, \ldots, \mathbf{u}_{n_t\Delta t}^{(i)} \right]$$

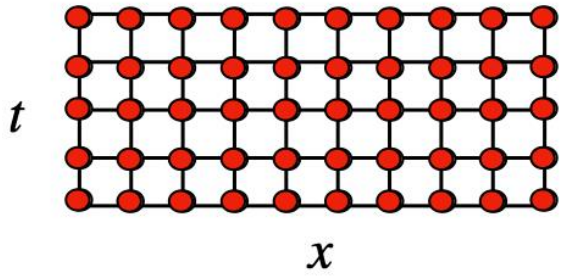
$$\mathcal{T}_{z,i} = \left[\mathbf{z}_0^{(i)}, \mathbf{z}_{1\Delta t}^{(i)}, \mathbf{z}_{2\Delta t}^{(i)}, \ldots, \mathbf{z}_{n_t\Delta t}^{(i)} \right]$$

Comparison

Vanilla - NO

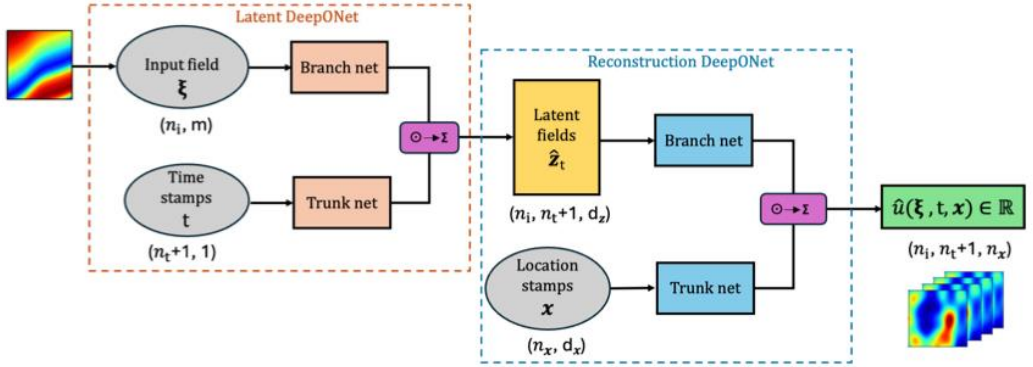


$$u = \sum_{i=1}^{p} Br_i(\xi) \cdot Tr_i(t, x)$$

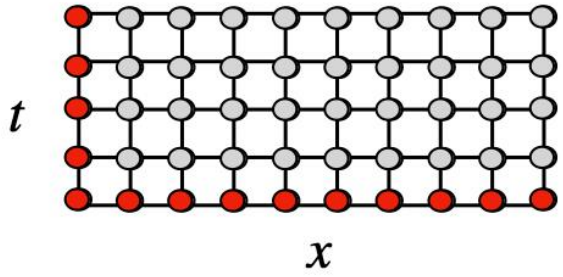


Trunk Network Evaluation

Latent - NO



$$u = \sum_{j=1}^{p} Br_j^r \left(\sum_{i=1}^{q} Br_i^l(\xi) \cdot Tr_i^l(t) \right) Tr_j^r(x)$$



Trunk Network Evaluation

Stove-Burner Simulation

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + s(x, y, r, a, \text{shape}),$$

PDE

$t \in (0, 1], \quad (x, y) \in \Omega = [-2, 2] \times [-2, 2],$

$D = 1,$

ICs: $u(0, x, y) = 0, \quad$ for all $(x, y) \in \Omega,$

BCs: $u(t, x, y) = 0, \quad$ for all $(x, y) \in \partial\Omega, \ t \in (0, 1]$

Input Function

$s(x, y, r, a, \text{shape})$

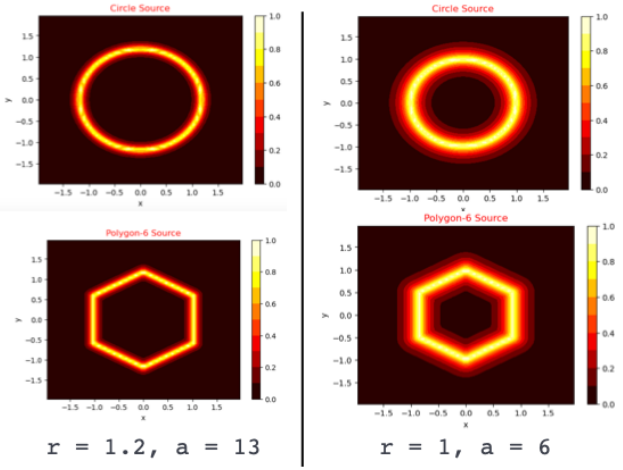
$r \sim \mathcal{U}(0.75, 1.25) \rightarrow$ Burner size (radius, side length, etc.)

$a \sim \mathcal{U}(5, 15) \rightarrow$ Burner Intensity factor

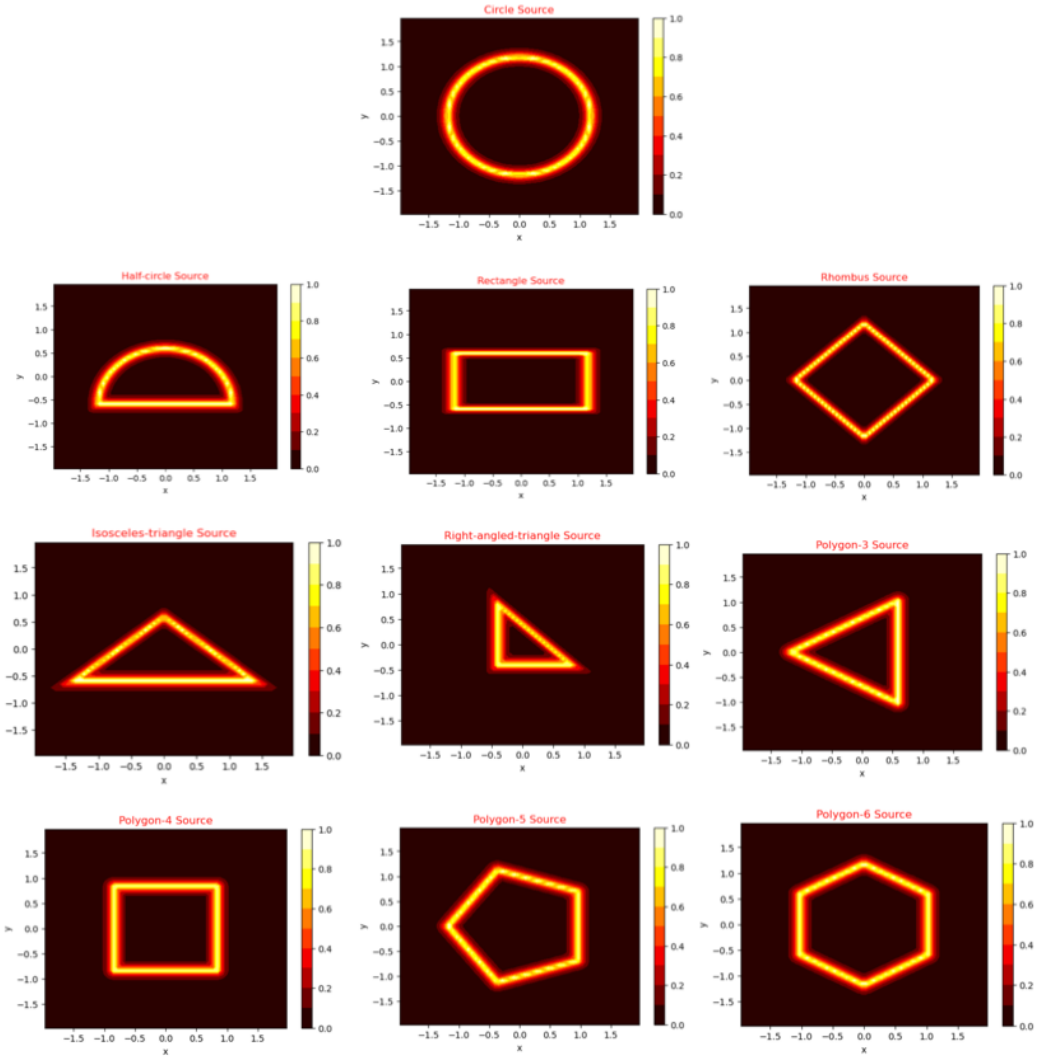
Operator

$\mathcal{G}_{\boldsymbol{\theta}} : s(x, y, r, a, \text{shape}) \rightarrow u(t, x)$

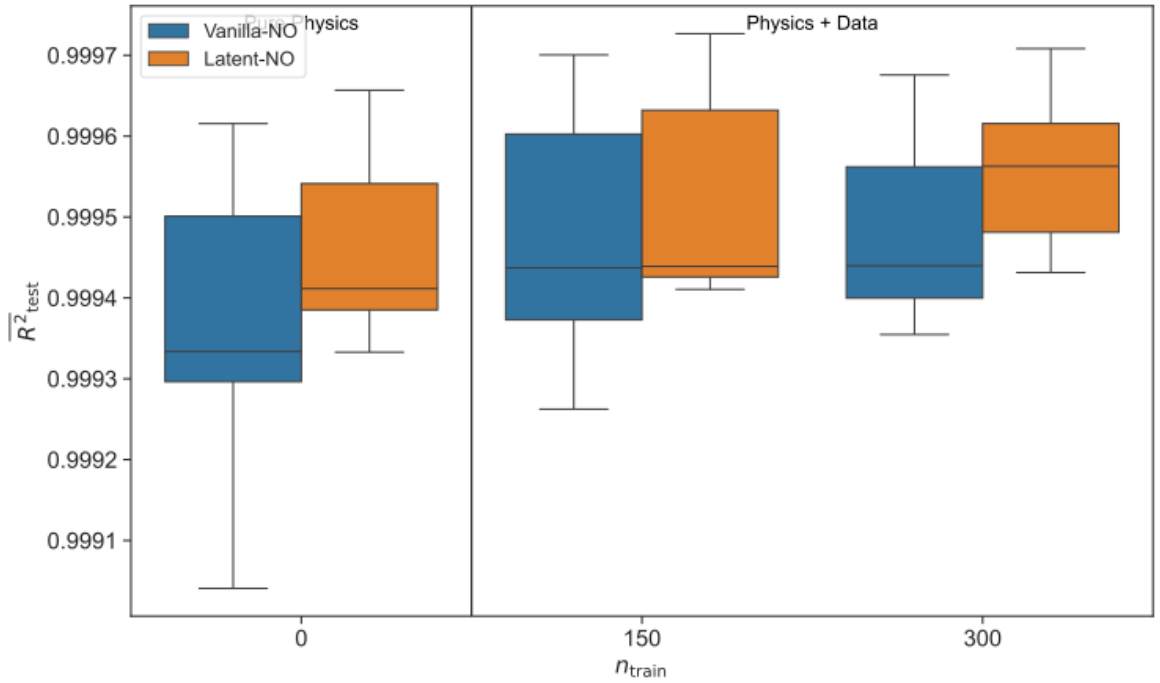
Varying Intensities

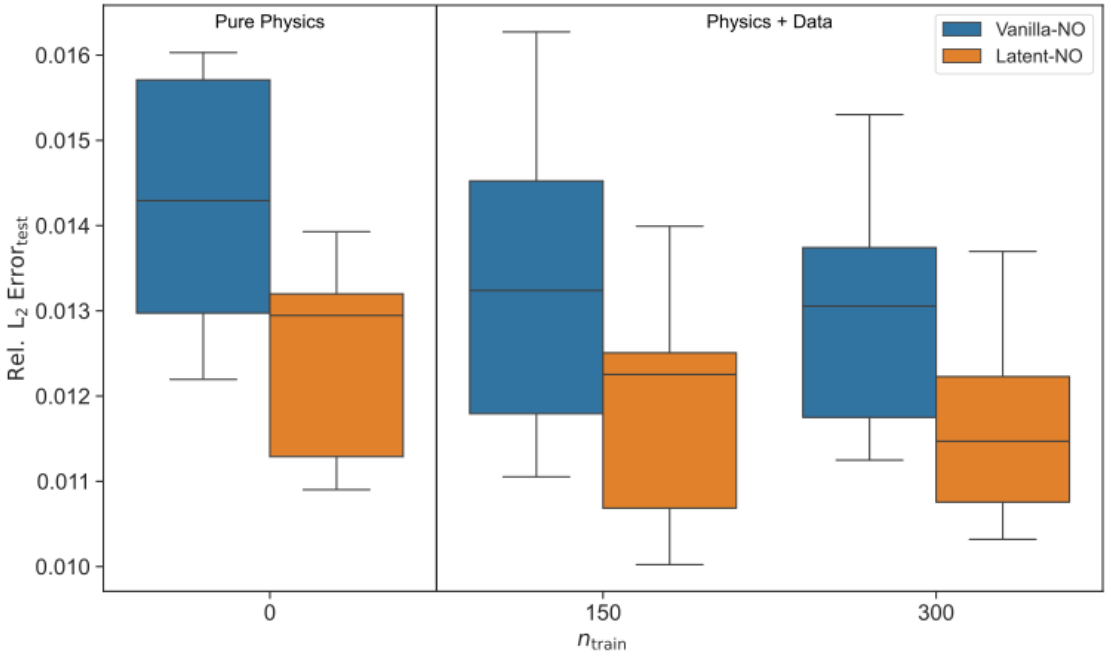


Different Burners

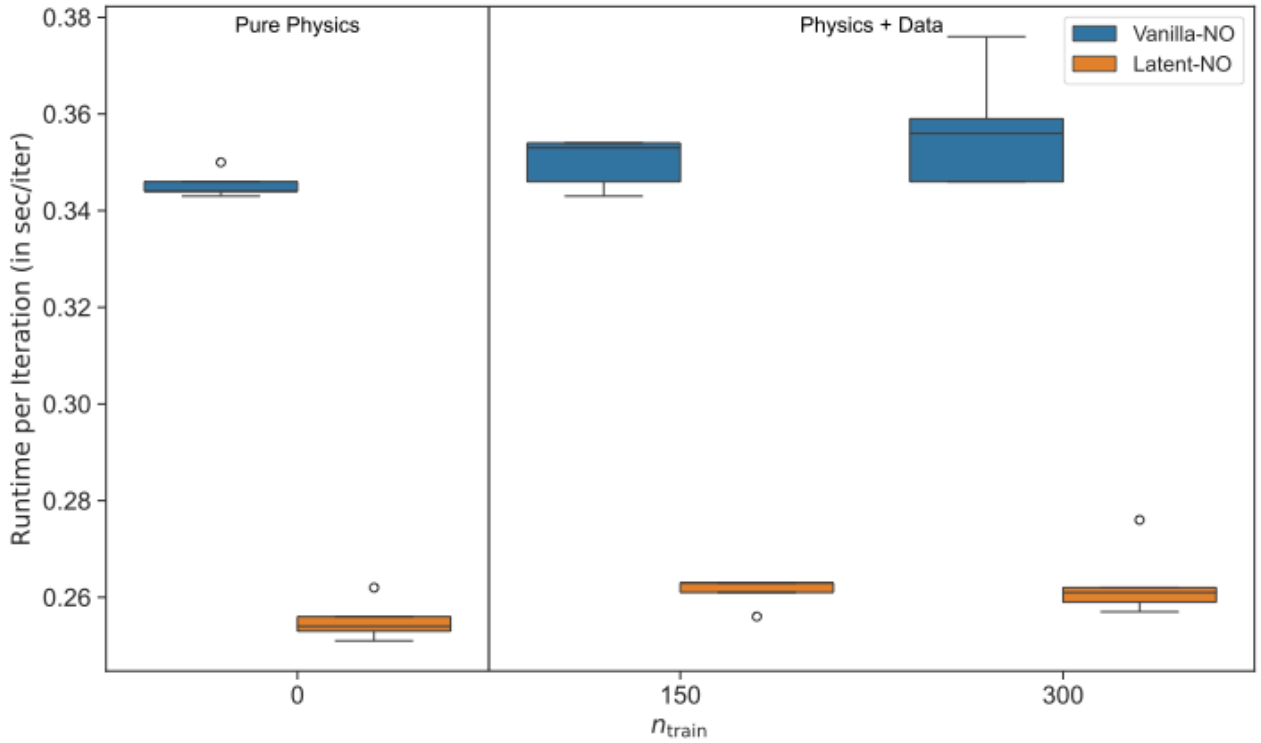


Stove-Burner Simulation





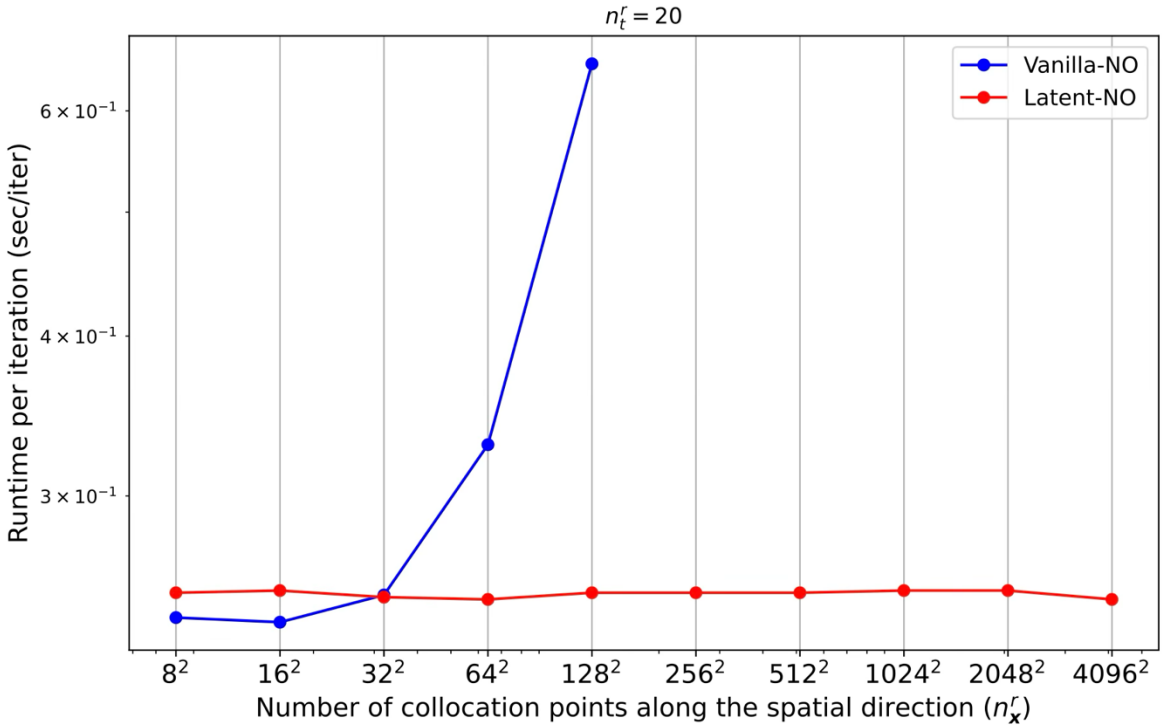
Stove-Burner Simulation

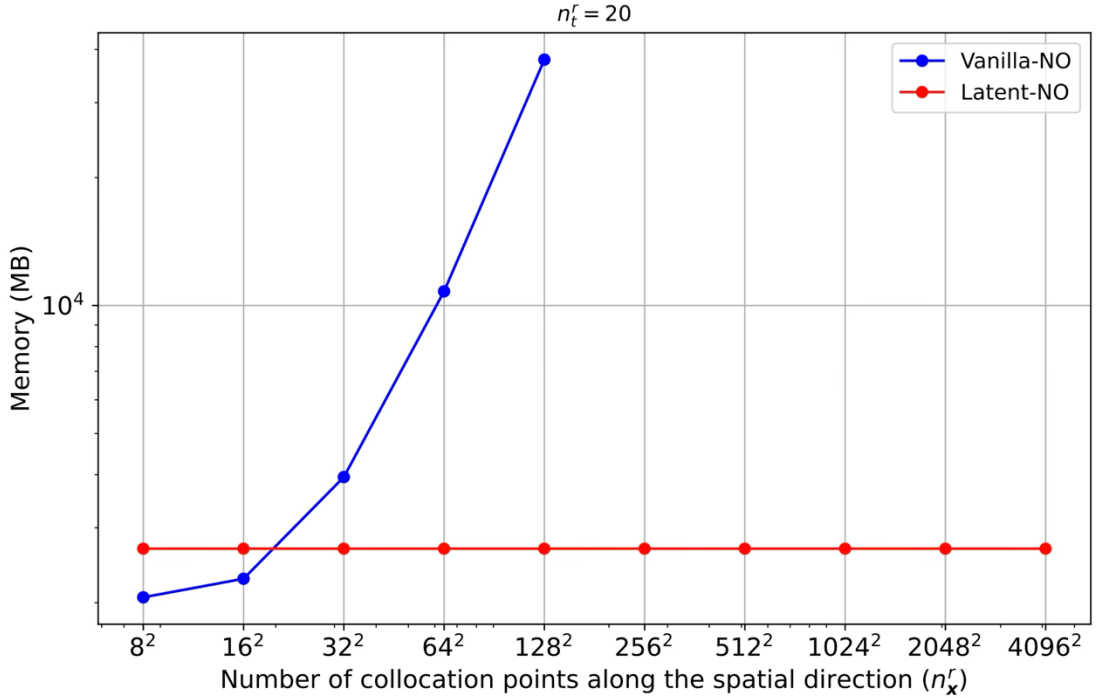


Our method achieves better accuracy with significantly reduced training time

Stove-Burner Simulation

Runtime and memory for different discretization sizes along the spatial direction.



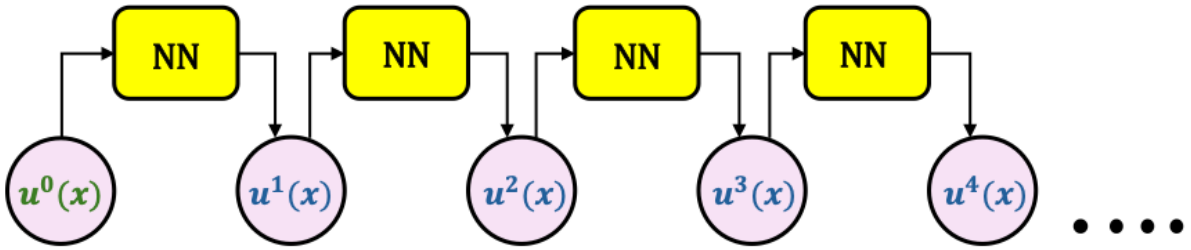


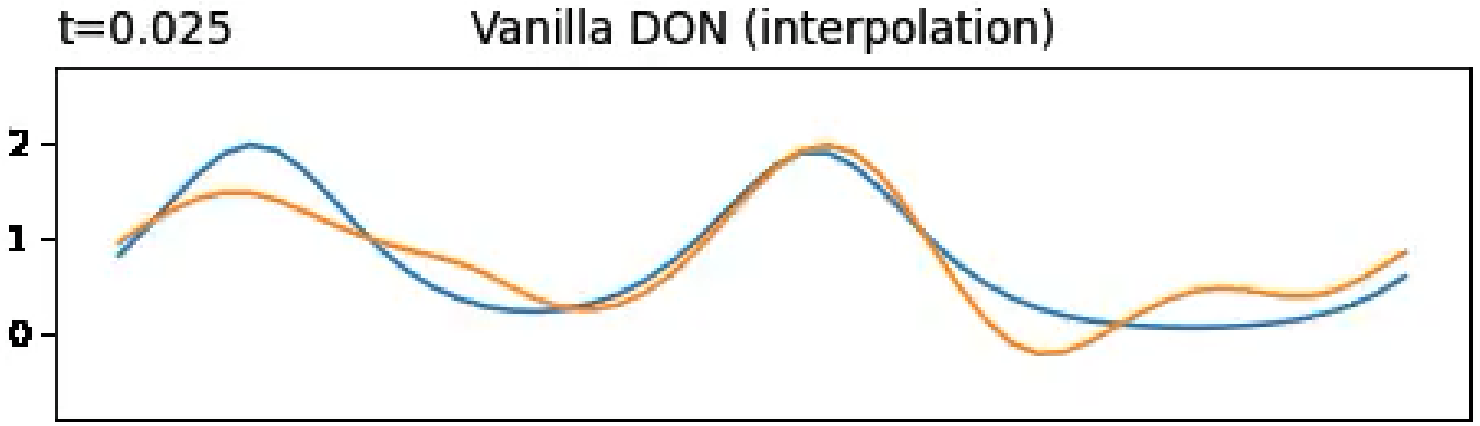
Other Shortcomings in Neural Operators

Fails in prediction of long-time horizons.

KdV equation: $u_t - \eta u u_x + \gamma u_{xxx} = 0$

Learning Task: $u(x, t = 0) \rightarrow u(x, t)$,





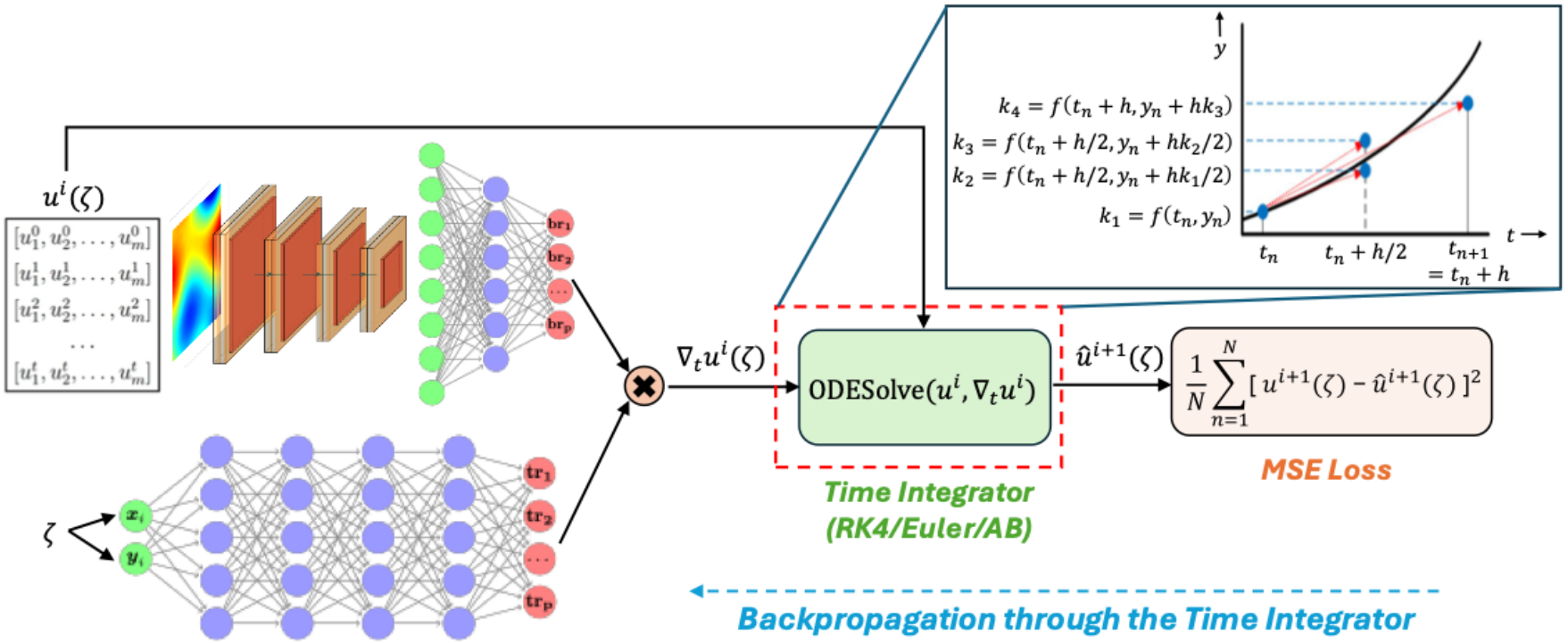
Our Proposed framework

TI-DeepONet: Learnable Time Integration for Stable Long-Term Extrapolation

Dibyajyoti Nayak
Department of Civil and Systems Engineering
Johns Hopkins University
Baltimore, MD, 21218
dnayak2@jh.edu

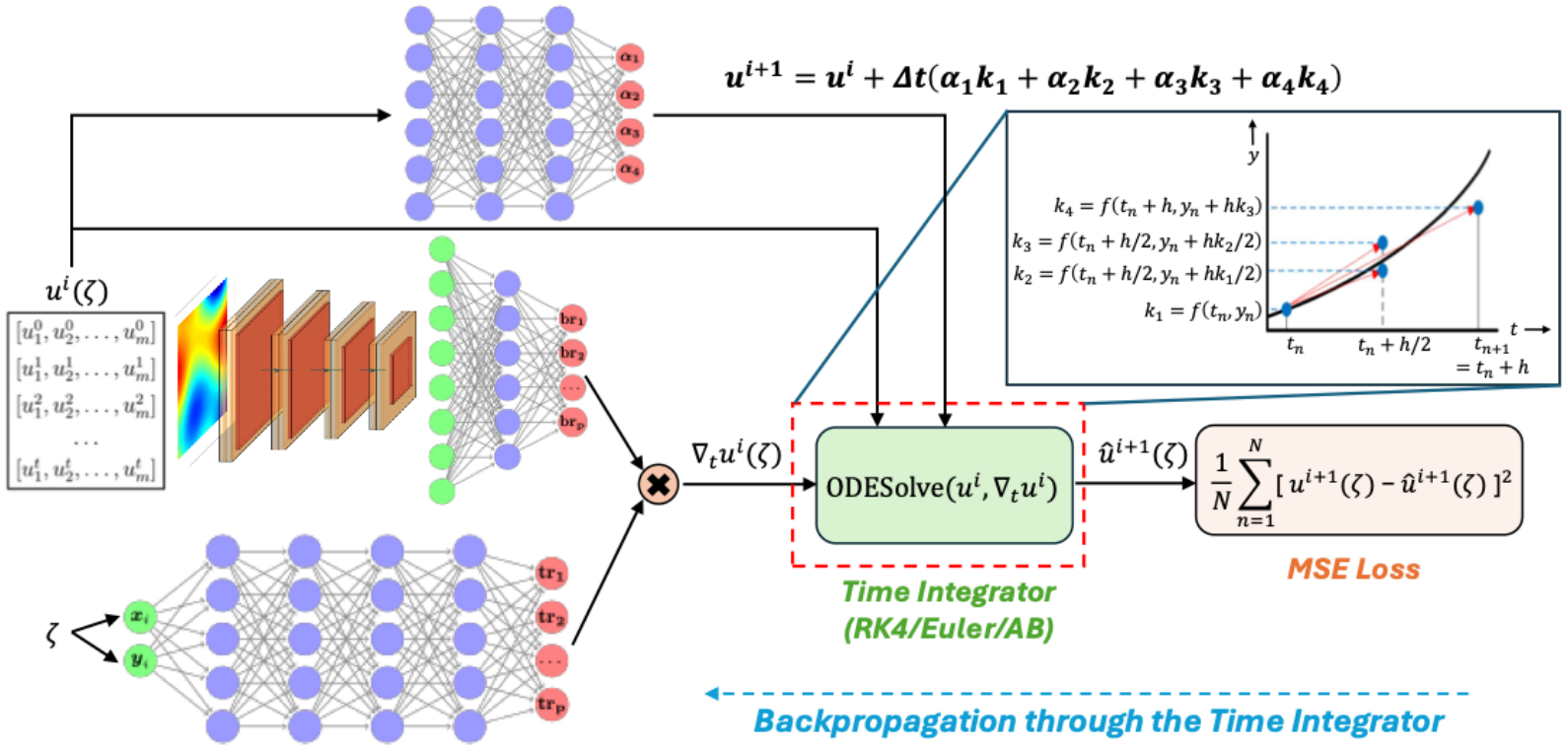
Somdatta Goswami
Department of Civil and Systems Engineering
Johns Hopkins University
Baltimore, MD, 21218
sgoswam4@jhu.edu

Time Integrated – DeepONet (TI-DeepONet)



While **training**, use **RK4**, however, while **inference** we could use **Adams–Bashforth or RK4**

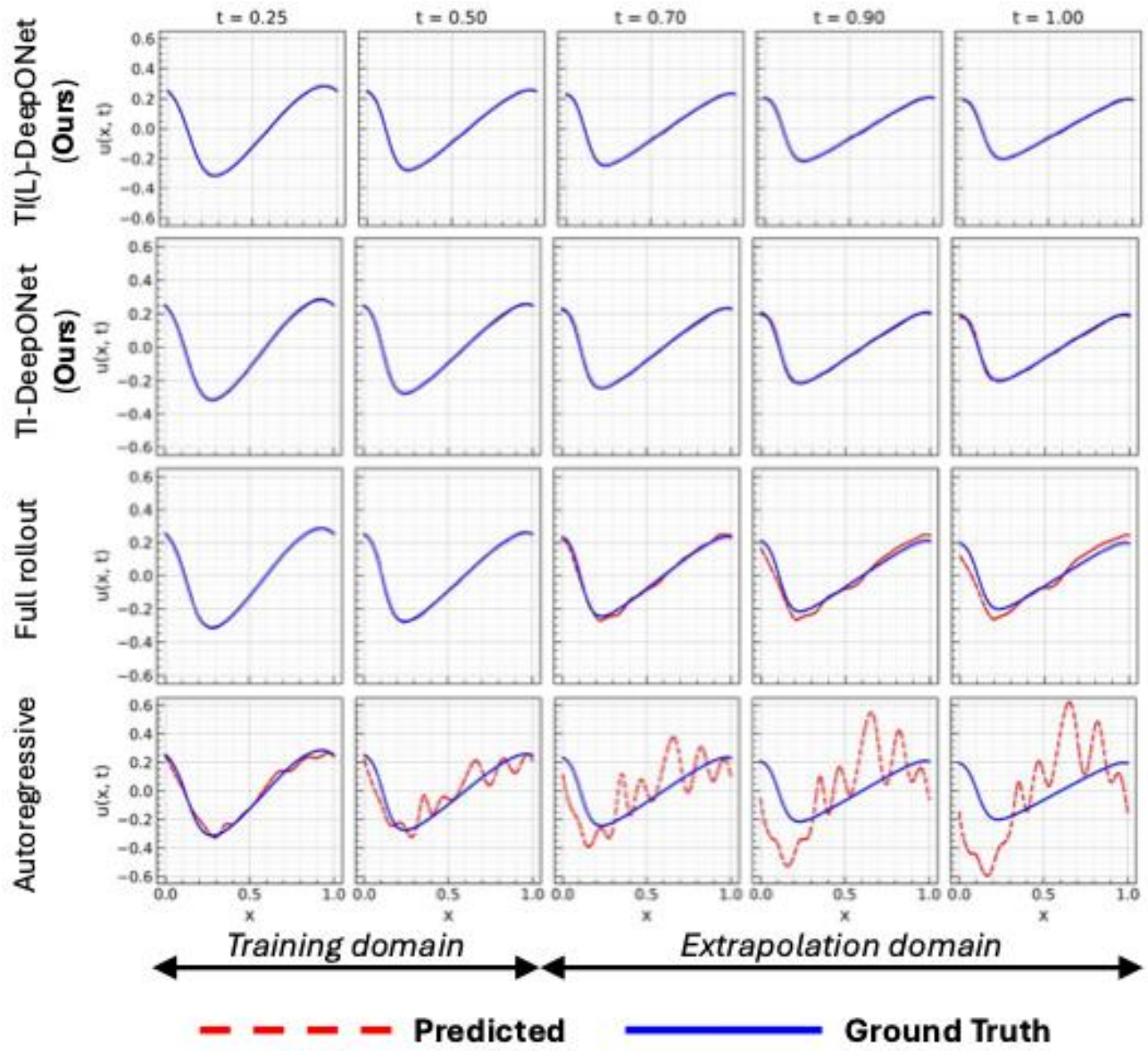
Learnable Time Integrated – DeepONet



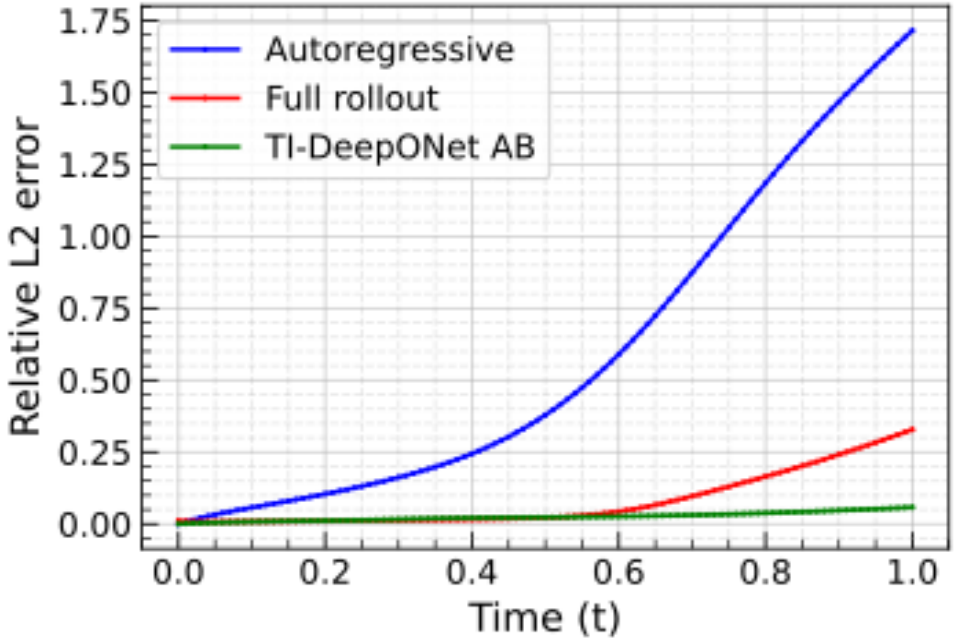
RK4 is employed in **Training** and **Inference.**

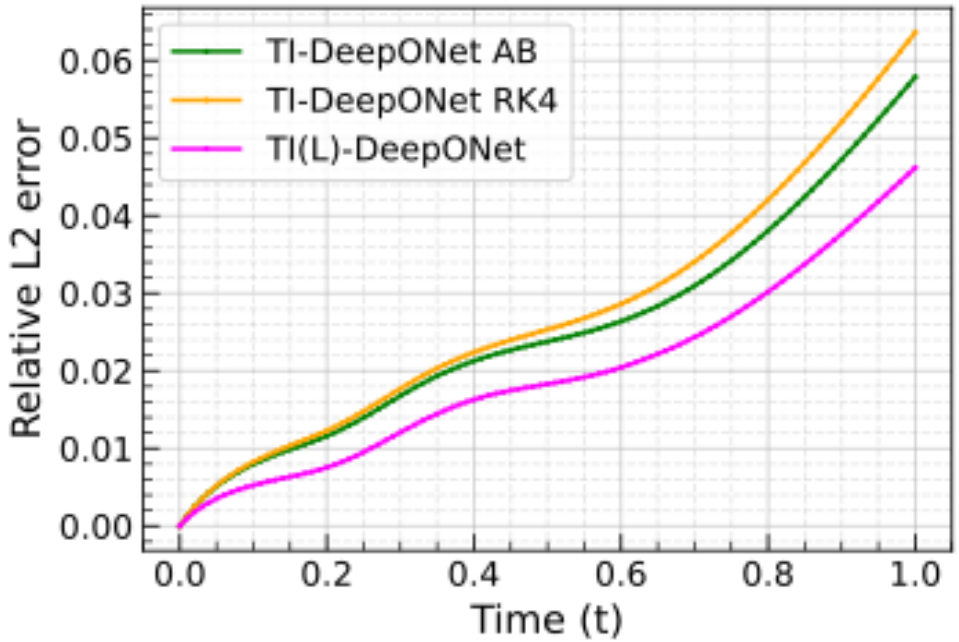
1D Burgers: Results

- Training: $t \in [0, 0.5]$, Inference: $t \in [0, 1]$
- Prediction of the solution profiles by the four different frameworks: (1) TI(L)-DeepONet, (2) TI-DeepONet, (3) Full rollout, and (4) Autoregressive
- Autoregressive accumulates errors early on and quickly deviates from the actual profile
- Full rollout performs well in training domain but starts incurring errors upon entering the extrapolation domain
- Both TI-DeepONet and TI(L)-DeepONet maintain a stable extrapolation resulting in an accurate and reliable prediction of solution state at later timesteps
- TI(L)-DeepONet slightly performs better due to its adaptivity to the local solution



1D Burgers: Results





- Autoregressive: Rapid error accumulation and resembles exponential error growth beyond $t = 0.5$.
- Full rollout: Performs well until $t = 0.5$ and then starts incurring errors.
- TI-DeepONet + AB2/AM3 inference: Stable and controlled error growth especially in extrapolation domain
- With the time integrator frameworks: TI(L)-DeepONet performs best followed by TI-DeepONet AB and then TI-DeepONet RK4

Results for 5 Independent Runs

Problem	Method	Relative L_2 error			
		t+10Δt_e	t+20Δt_e	t+40Δt_e	T^*
Burgers' (1D)	TI(L)-DeepONet	**0.019±0.003**	**0.023±0.003**	**0.036±0.004**	**0.044±0.005**
	TI-DeepONet AB	0.031±0.004	0.037±0.005	0.057±0.008	0.070±0.011
	Full Rollout	0.043±0.002	0.095±0.004	0.247±0.028	0.336±0.053
	Autoregressive	0.710±0.089	1.004±0.144	1.556±0.206	1.768±0.227
KdV (1D)	TI(L)-DeepONet	**0.054±0.019**	**0.065±0.027**	**0.075±0.031**	**0.111±0.051**
	TI-DeepONet AB	0.086±0.026	0.108±0.034	0.129±0.043	0.183±0.063
	Full Rollout	0.776±0.0004	0.716±0.0005	0.719±0.0005	0.795±0.0007
	Autoregressive	0.823±0.073	0.886±0.064	0.922±0.069	0.968±0.083
Burgers' (2D)	TI(L)-DeepONet	**0.111±0.002**	**0.121±0.003**	**0.143±0.004**	**0.155±0.004**
	TI-DeepONet AB	0.121±0.002	0.133±0.002	0.157±0.003	0.169±0.003
	Full Rollout	0.131±0.007	0.194±0.014	0.357±0.035	0.453±0.049
	Autoregressive	0.503±0.017	0.590±0.024	0.783±0.052	0.894±0.075

Can Neural Operators Help Identify Systems?

Learning Hidden Physics and System Parameters with Deep Operator Networks

Vijay Kag[a,1], Dibakar Roy Sarkar[b,1], Birupaksha Pal[a,*], Somdatta Goswami[b]

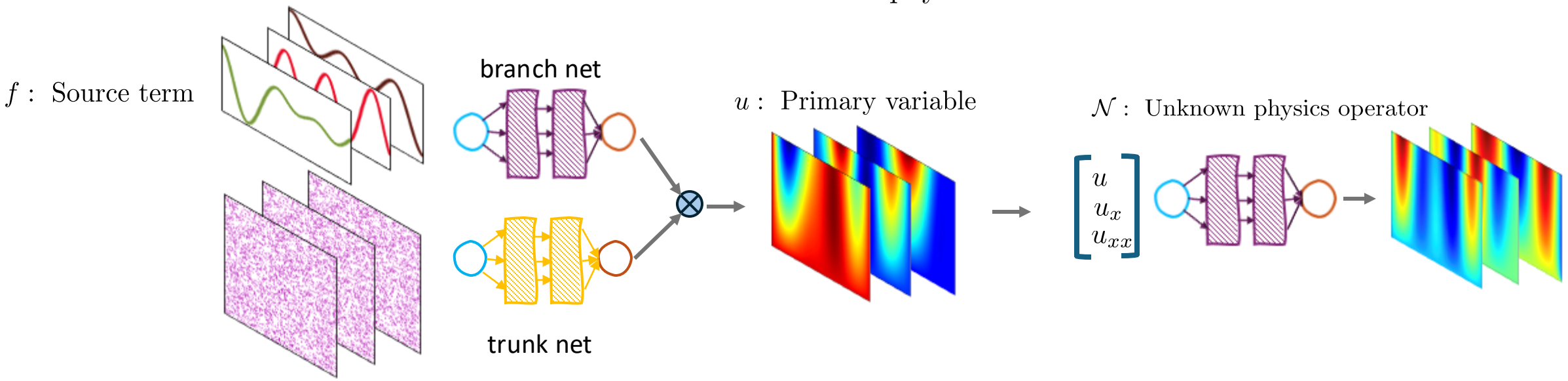
[a] *Robert Bosch Research and Technology Center Bangalore, 123 Industrial layout, Karnataka 560095, India*
[b] *John Hopkins Whiting School of Engineering, Baltimore 3400 N Charles St, MD 21218, United States*




#1: The Deep Hidden Physics Operator (DHPO) network

$$\frac{\partial u}{\partial t} = \mathcal{N}(t, x, u, u_x, u_{xx}, \ldots) + f(x)$$

u : Primary variable
f : Source term
$\mathcal{N}$: Unknown physics



Loss, $\mathcal{L} = \mathcal{L}_{\text{pde}} + \mathcal{L}_{\text{bc}} + \mathcal{L}_{\text{ic}} + \mathcal{L}_{\text{data}}$

Consider Burger's equation

$$\mathcal{L}_{\text{pde}} = ||u_t^{\text{NO}} + \mathcal{N}^{\text{DHPO}} - f(x)||^2$$

$$\mathcal{L}_{\text{bc}} = ||u^{\text{NO}} - u^{\text{bc}}||^2$$

$$\mathcal{L}_{\text{ic}} = ||u^{\text{NO}} - u^{\text{ic}}||^2$$

$$\mathcal{L}_{\text{data}} = ||u^{\text{NO}} - u^{\text{data}}||^2$$

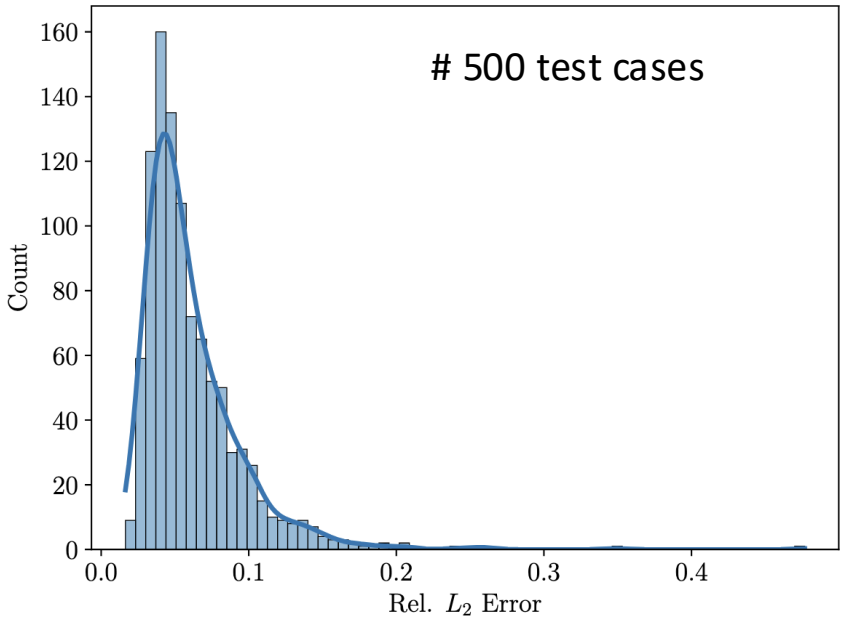
Results: Physics Discovery

Burger's problem:

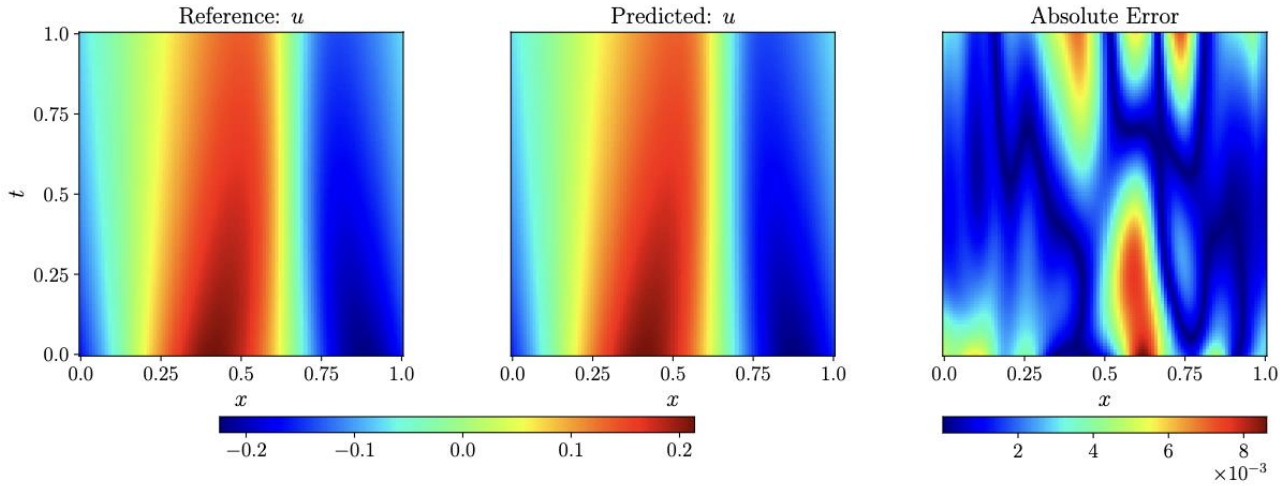
$$\frac{\partial u}{\partial t}(x,t) = \overbrace{\nu \frac{\partial^2 u}{\partial x^2}(x,t) - u\frac{\partial u}{\partial x}(x,t)}^{\mathcal{N}(u, u_x, u_{xx})} \text{ on } \Omega : (x,t) \in [0,1]^2,$$

$$\text{IC: } u(x,0) = f(x),$$

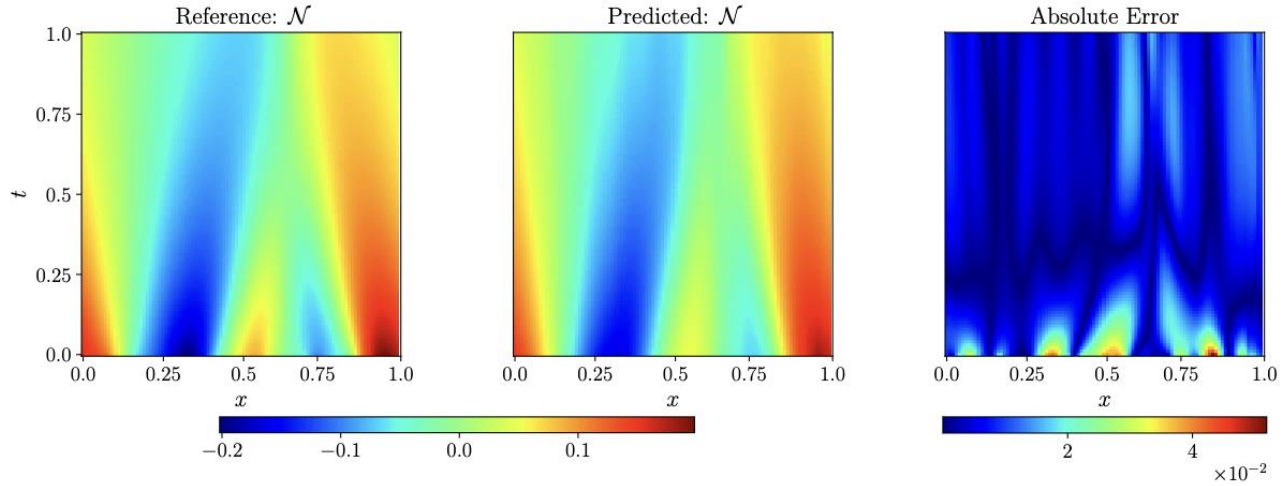
$$\text{BC: } u(0,t) = u(1,t) \text{ and } \frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(1,t),$$



Average relative L_2 error of $\mathcal{N}(u, u_x, u_{xx})$ $\mathcal{O}(10^{-2})$



(a) Sample 1: solution field accuracy comparison, relative L_2 error = 0.02134.



(b) Sample 1: hidden physics solution comparison, relative L_2 error = 0.118278.

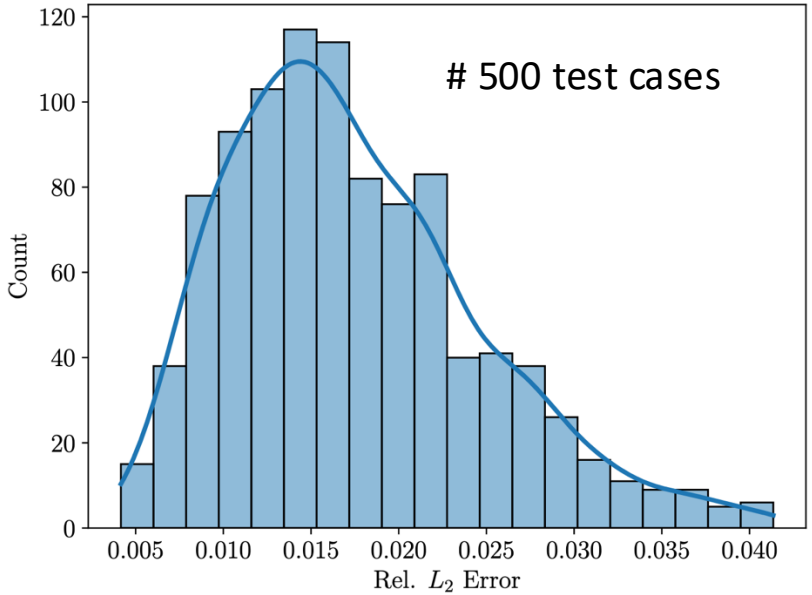
Results: Physics Discovery

Reaction diffusion problem:

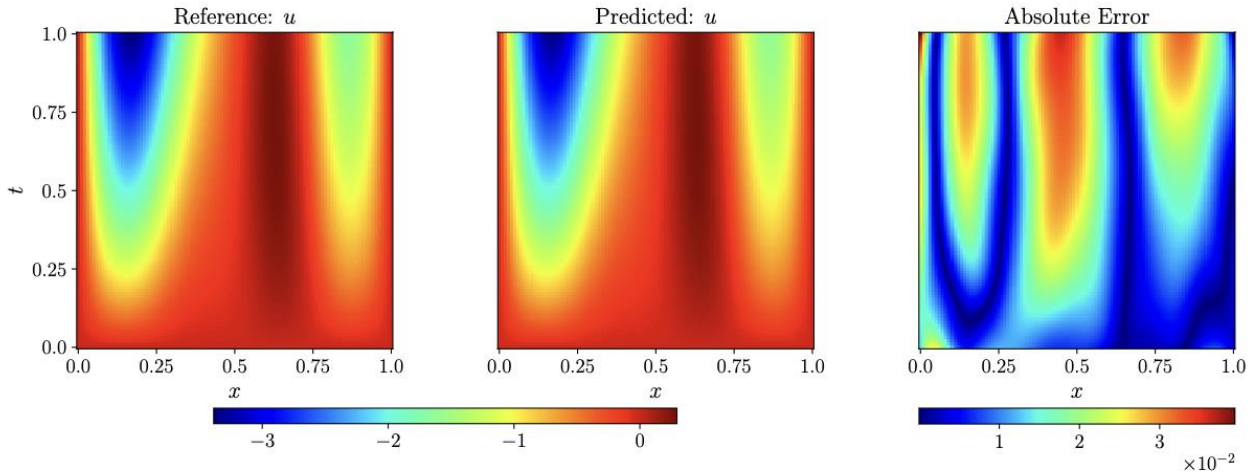
$$\frac{\partial u}{\partial t}(x,t) = \overbrace{D\frac{\partial^2 u}{\partial x^2}(x,t) + Ku^2(x,t)}^{\mathcal{N}(u,u_x,u_{xx})} + \boxed{f(x)} \text{ on } \Omega : (x,t) \in [0,1]^2,$$

$$IC : u(x,0) = 0,$$

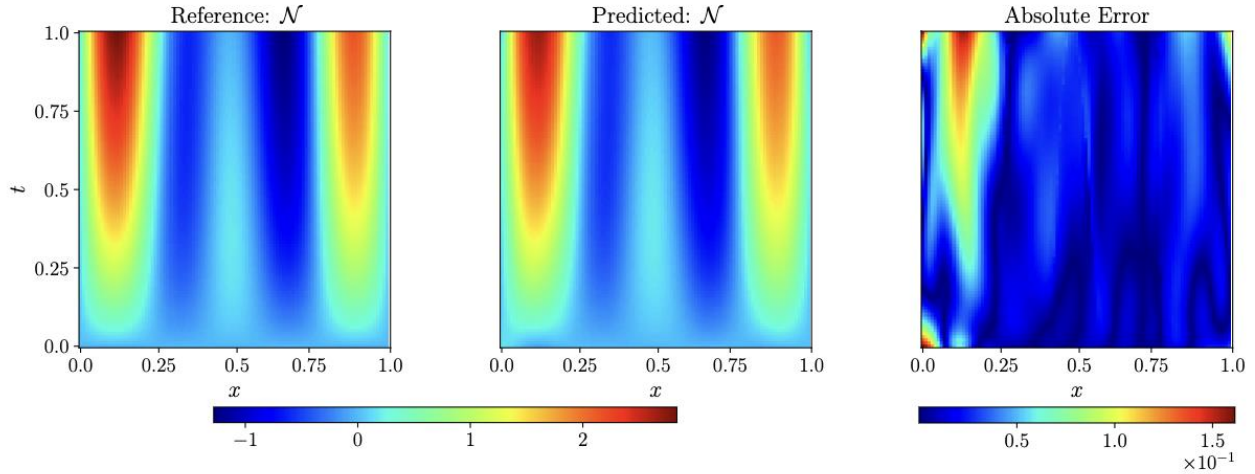
$$BC : u(0,t) = u(1,t) = 0,$$



Average relative L_2 error of $\mathcal{N}(u, u_x, u_{xx})$ $\mathcal{O}(10^{-2})$



(a) Sample 1: solution field accuracy comparison, relative L_2 error = 0.01639.



(b) Sample 1: hidden physics solution comparison, relative L_2 error = 0.03949.

#2: Neural Operator for System Parameter Identification

Consider Burger's equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

u : Primary variable

ν : Viscosity

Step 1: Learn solution operator

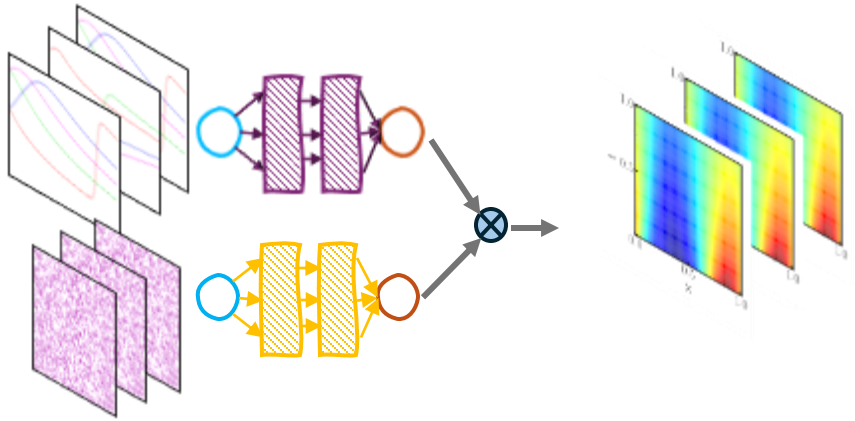
Step 2: Learn parameter to solution operator

Loss, $\mathcal{L} = \mathcal{L}_{\text{pde}} + \mathcal{L}_{\text{sensor data}}$

$$\mathcal{L}_{\text{pde}} = ||u_t^{\text{NO}} + u^{\text{NO}} u_x^{\text{NO}} - \nu^{\text{INO}} u_{xx}^{\text{NO}}||^2$$

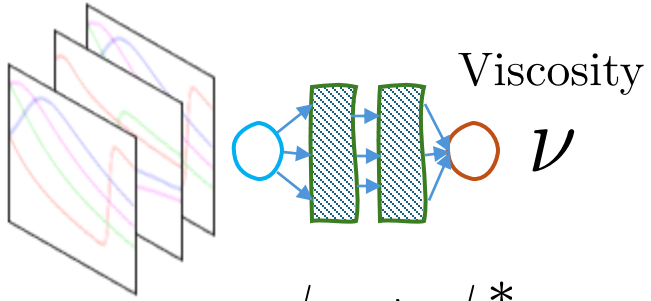
$$\mathcal{L}_{\text{sensor data}} = ||u^{\text{NO}} - u^{\text{data}}||^2$$

u_{data} : Sensor data u : Primary variable



$\theta \rightarrow \theta^* \rightarrow \theta^{**}$

Continual Learning



$\phi \rightarrow \phi^*$

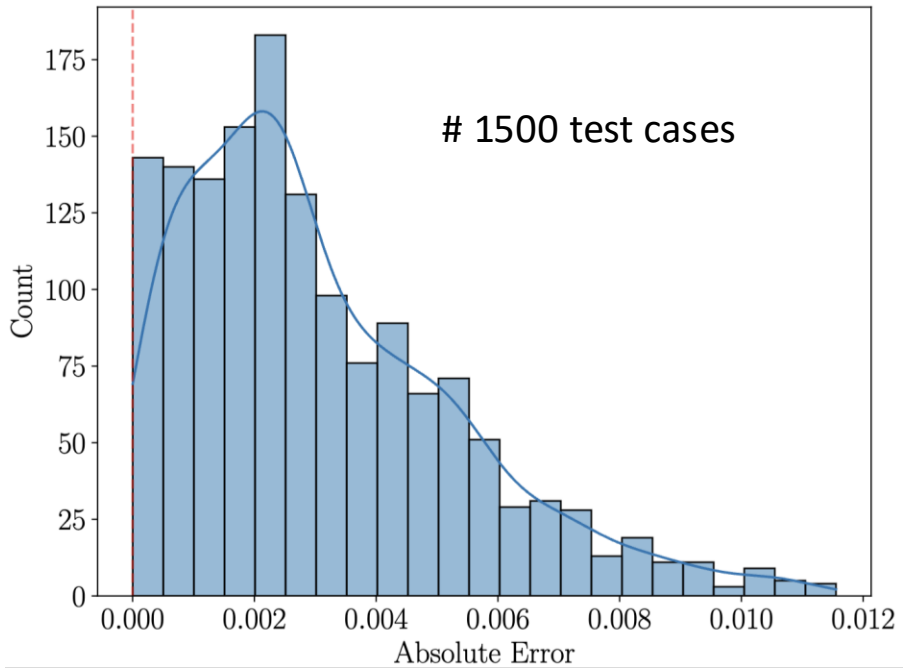
Results: Parameter Identification

Burger's problem:

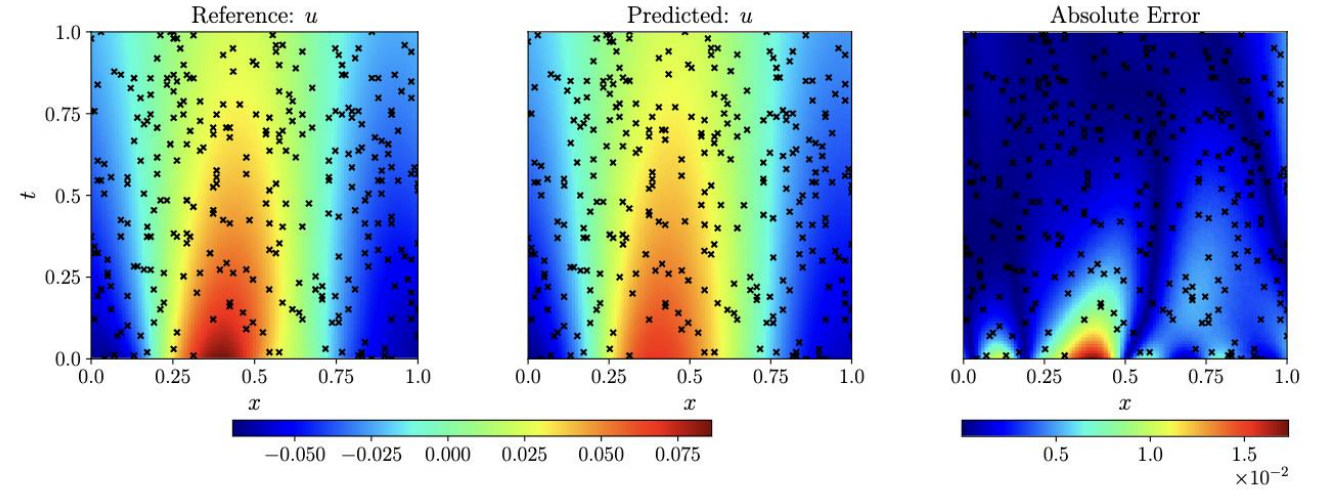
$$\frac{\partial u}{\partial t}(x,t) = \nu \frac{\partial^2 u}{\partial x^2}(x,t) - u\frac{\partial u}{\partial x}(x,t) \text{ on } \Omega : (x,t) \in [0,1]^2,$$

IC: $u(x,0) = f(x)$,

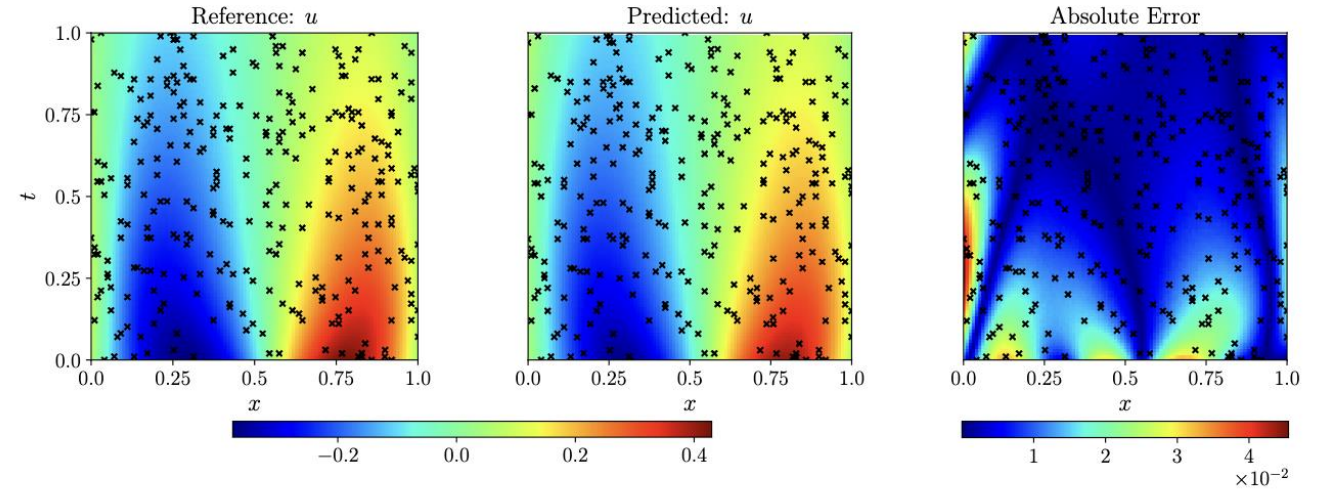
BC: $u(0,t) = u(1,t)$ and $\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(1,t)$,



Average absolute error of viscosity, : $\nu \quad \mathcal{O}(10^{-3})$



(a) Sample 1: Relative L_2 error of $u(x,t) = 0.092$. For this case, $\nu_{\text{true}} = 0.028$ while $\nu_{\text{predicted}} = 0.025$.



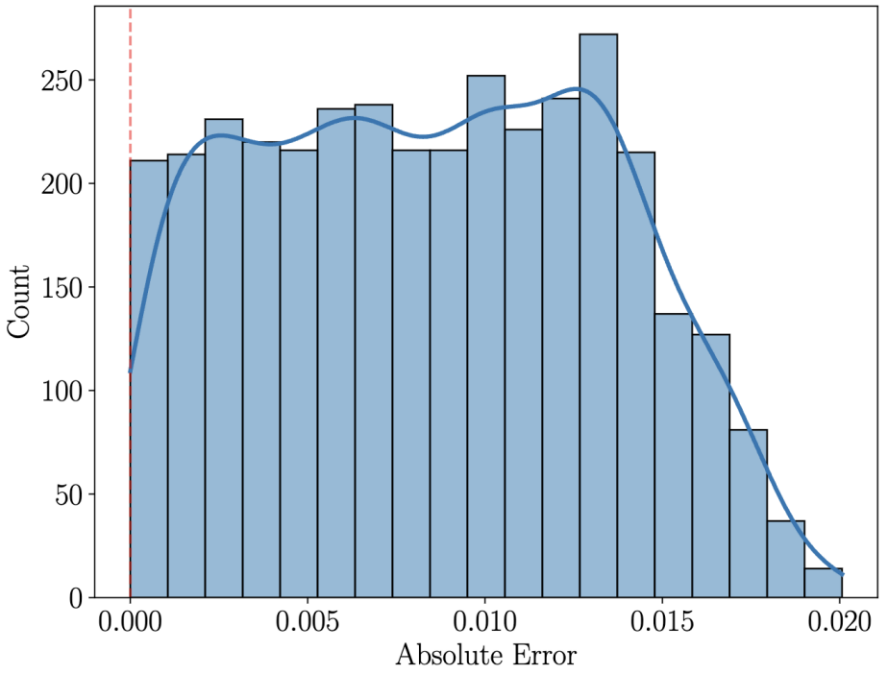
(b) Sample 2: Relative L_2 error of $u(x,t) = 0.061$. For this case, $\nu_{\text{true}} = 0.032$ while $\nu_{\text{predicted}} = 0.032$.

Results: Parameter Identification

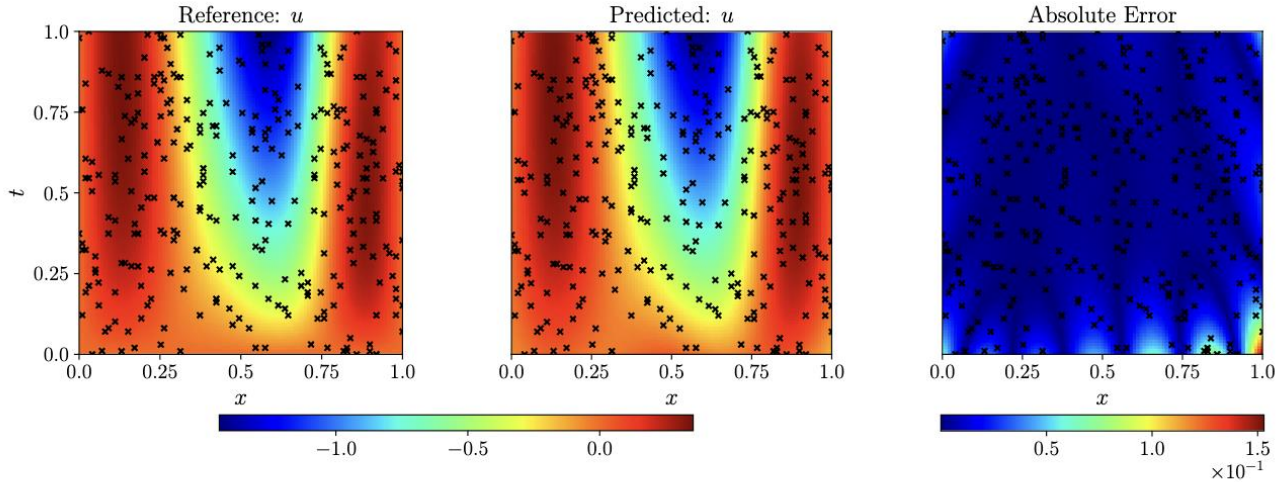
Reaction diffusion problem:

$$\frac{\partial u}{\partial t}(x,t) = D\frac{\partial^2 u}{\partial x^2}(x,t) + Ku^2(x,t) + f(x) \text{ on } \Omega : (x,t) \in [0,1]^2$$
$$IC : u(x,0) = 0,$$
$$BC : u(0,t) = u(1,t) = 0,$$

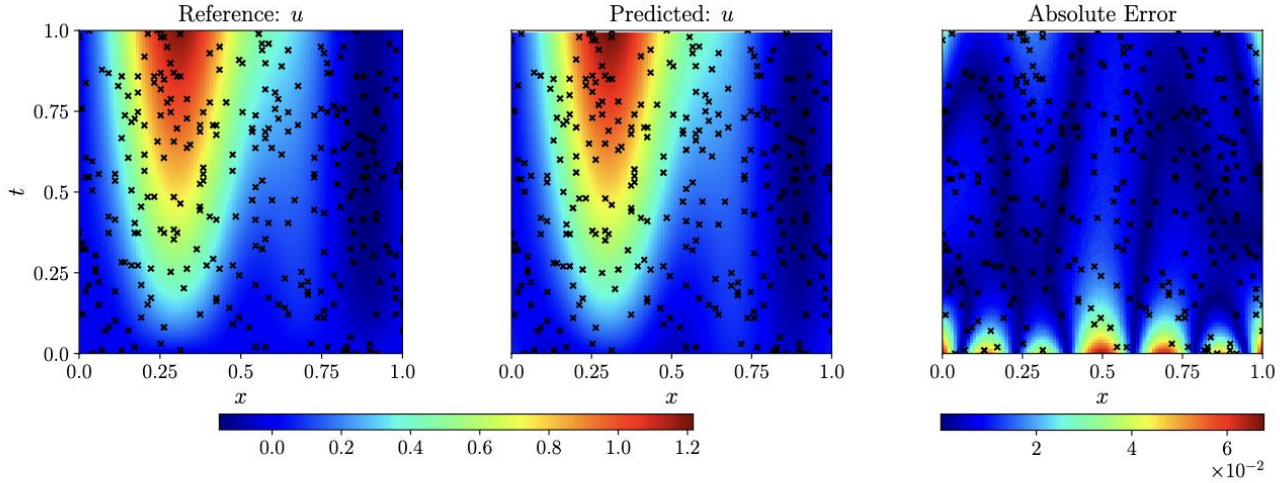
1500 test cases



Average absolute error of diffusion coefficient, $: D \quad \mathcal{O}(10^{-3})$

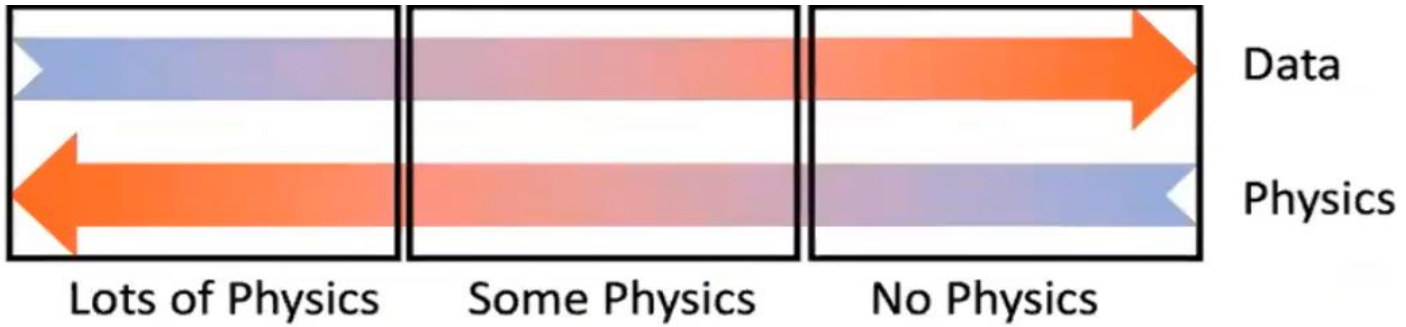


(a) Sample 1: Relative L_2 error of $u(x,t) = 0.027$. For this case, $D_{\text{true}} = 0.027$ while $D_{\text{predicted}} = 0.027$.



(b) Sample 2: Relative L_2 error of $u(x,t) = 0.029$. For this case, $D_{\text{true}} = 0.036$ while $D_{\text{predicted}} = 0.034$.

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.

For more work on similar topics:
https://sites.google.com/view/centrum-intelliphysics/home

The codes for all our work:
https://github.com/Centrum-IntelliPhysics

Centrum-IntelliPhysics

Overview | Repositories 9 | Projects | Pack

Centrum IntelliPhysics

Benchmark dataset: Phase Field fracture modeling in hyperelastic material with multiple cracks

Hyperelastic multi-crack response studied under cyclic loading simulated with FEniCS

This dataset contains 1000 finite-element simulations of a 2D square specimen with a circular cavity and eigh pre-cracks, subjected to tension, unloading, and compression using a large-deformation Neo-Hookean phas field formulation.

Full Dataset

Our complete collection comprises **1000** distinct Gaussian-random-field-driven boundary-condition realizations. The full dataset is available in JHU archive at: https://archive.data.jhu.edu/dataset.xhtml?persistentId=doi:10.7281/T1XFF19O

Johns Hopkins Libraries | Johns Hopkins Research Data Repository

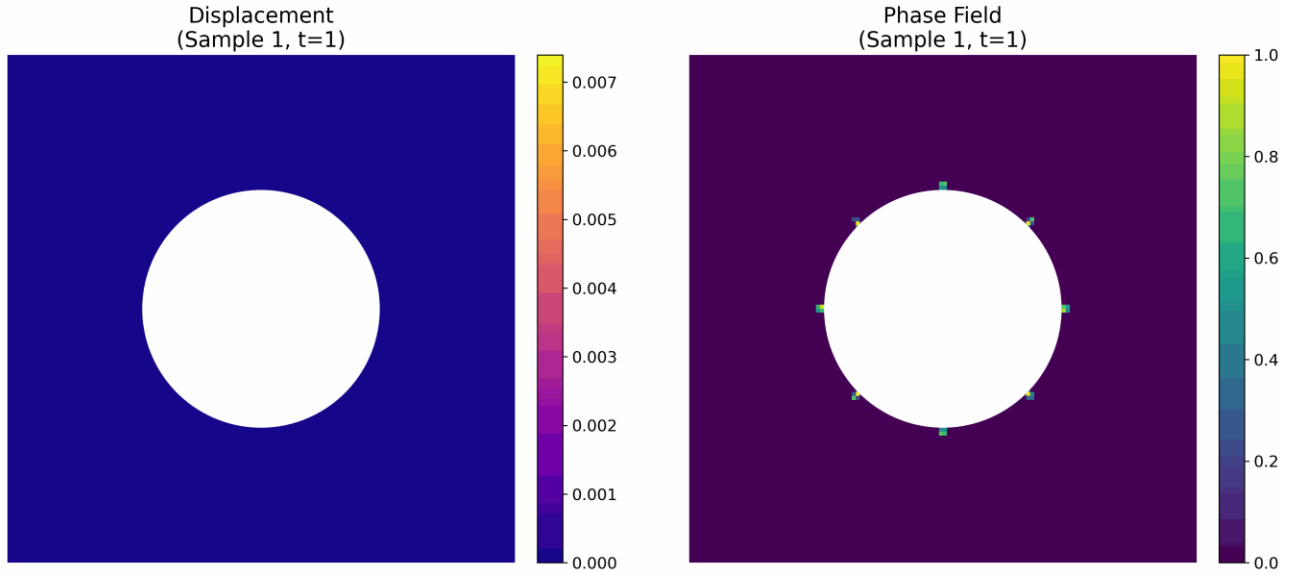
Johns Hopkins Research Data Repository

(JHU Data Services)

Johns Hopkins Research Data Repository >

Data and code associated with: Phase-Field Fracture Simulation Dataset: Hyperelastic Multi-Crack Response Under Loading and Unloading

Version 2.0



Acknowledgement





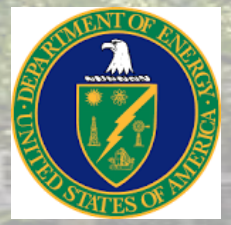








Funding



Thank you!