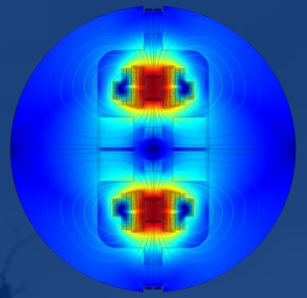
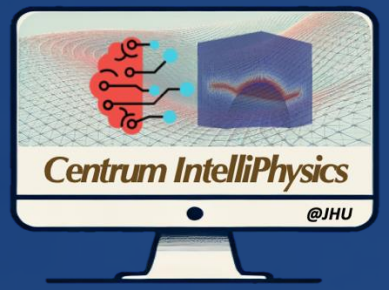


ICCE 2024



JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Employing Machine Learning Approaches to solve PDEs in "Mechanics"

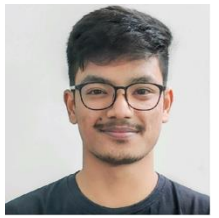
Somdatta Goswami
Assistant Professor, Civil and Systems Engineering

October 1, 2024

5th International Conference on Computational Engineering (ICCE 2024)

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

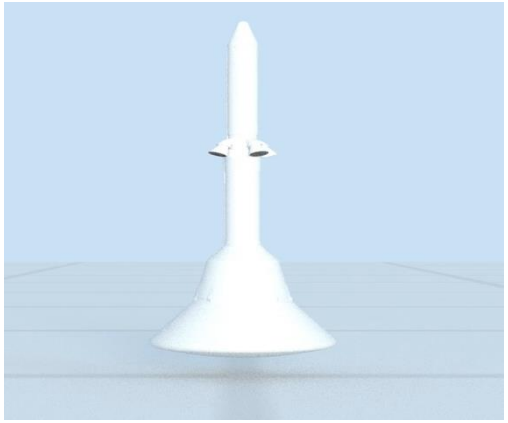
Can represent the Processes of Nature

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

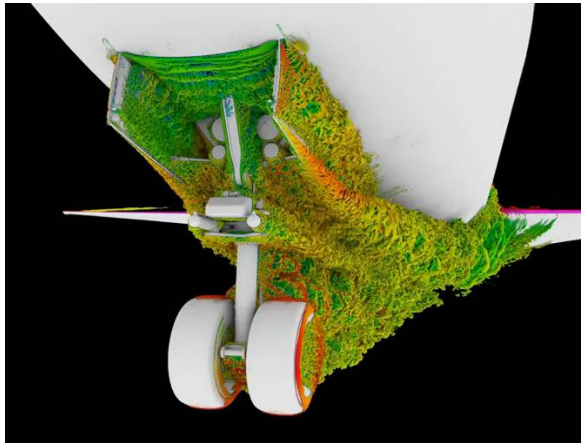
  To model earthquake: $m\frac{d^2u}{dt^2} + \mathrm{k}\frac{du}{dt} + F_0 = 0$

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

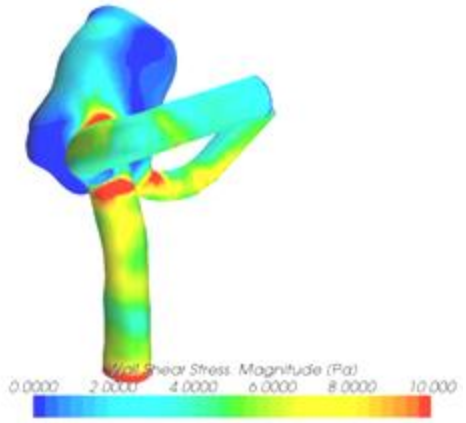
- Computational Mechanics helps us simulate these equations.



Simulation of Orion Spacecraft Launch Abort System (NASA Ames)

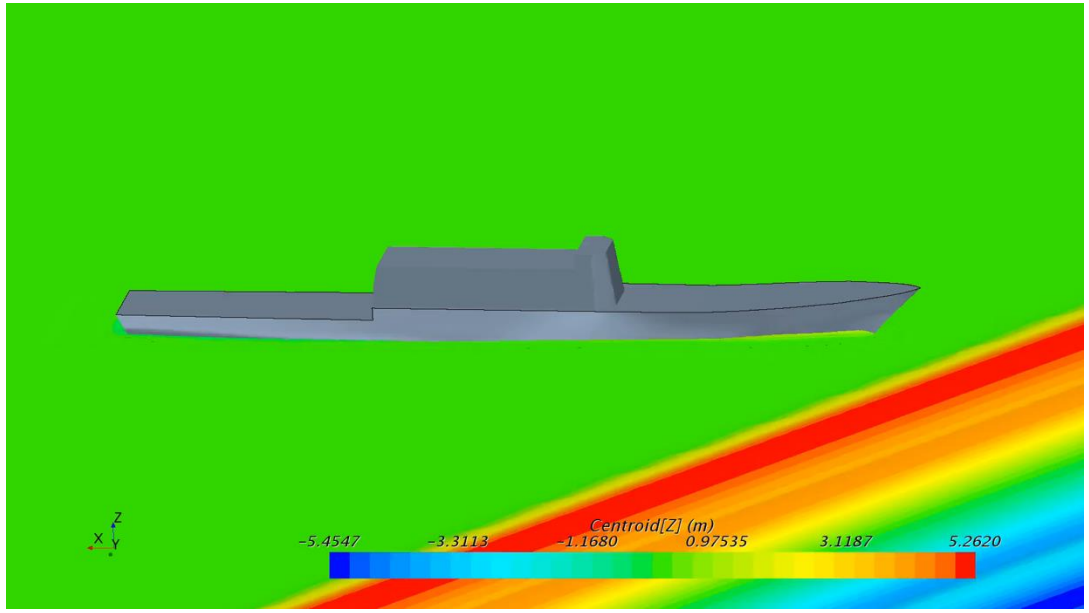


Detailed flow around an Aircraft's landing gear (NASA Ames)

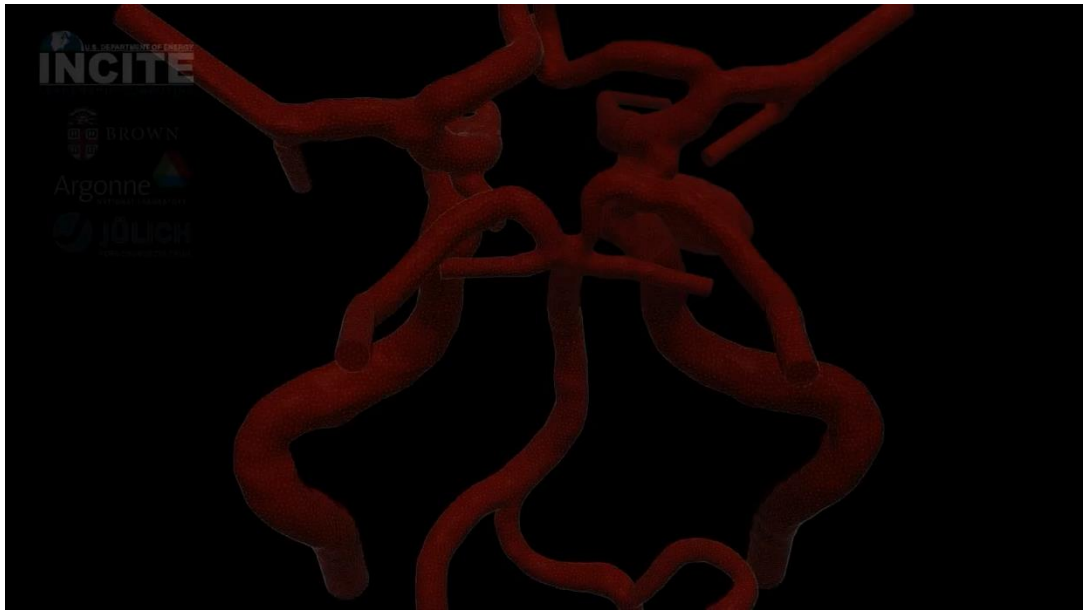


CFD Simulation of a Patient-Specific Intracranial Aneurysm

Simulation Real Scenarios



Courtesy: MIT, Michael Triantafyllou

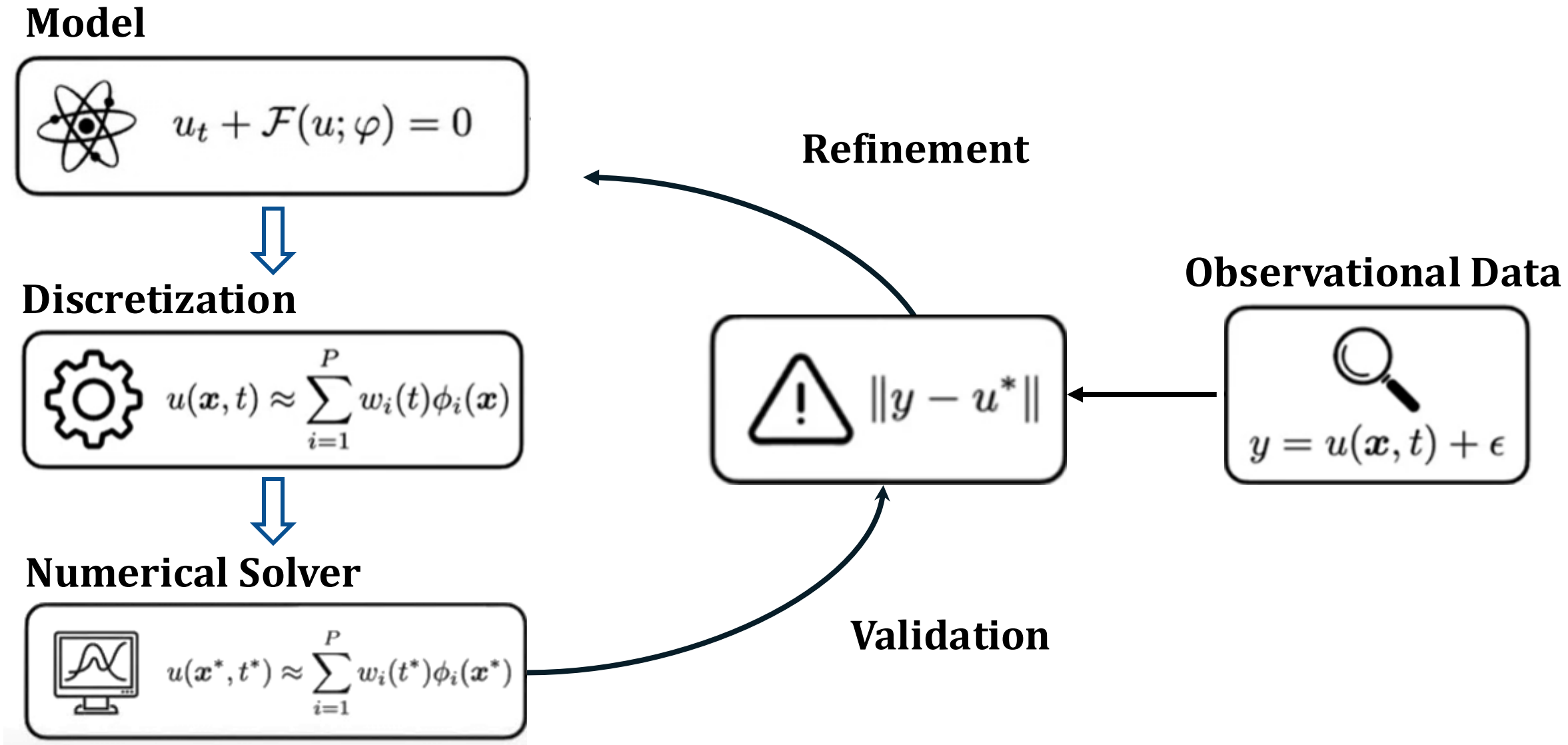


Courtesy: Argonne National Lab

Are we really mimicking real-world?

Courtesy: Boston Children's Hospital,
J. Marsden

Traditionally Solving Computational Mechanics Problems



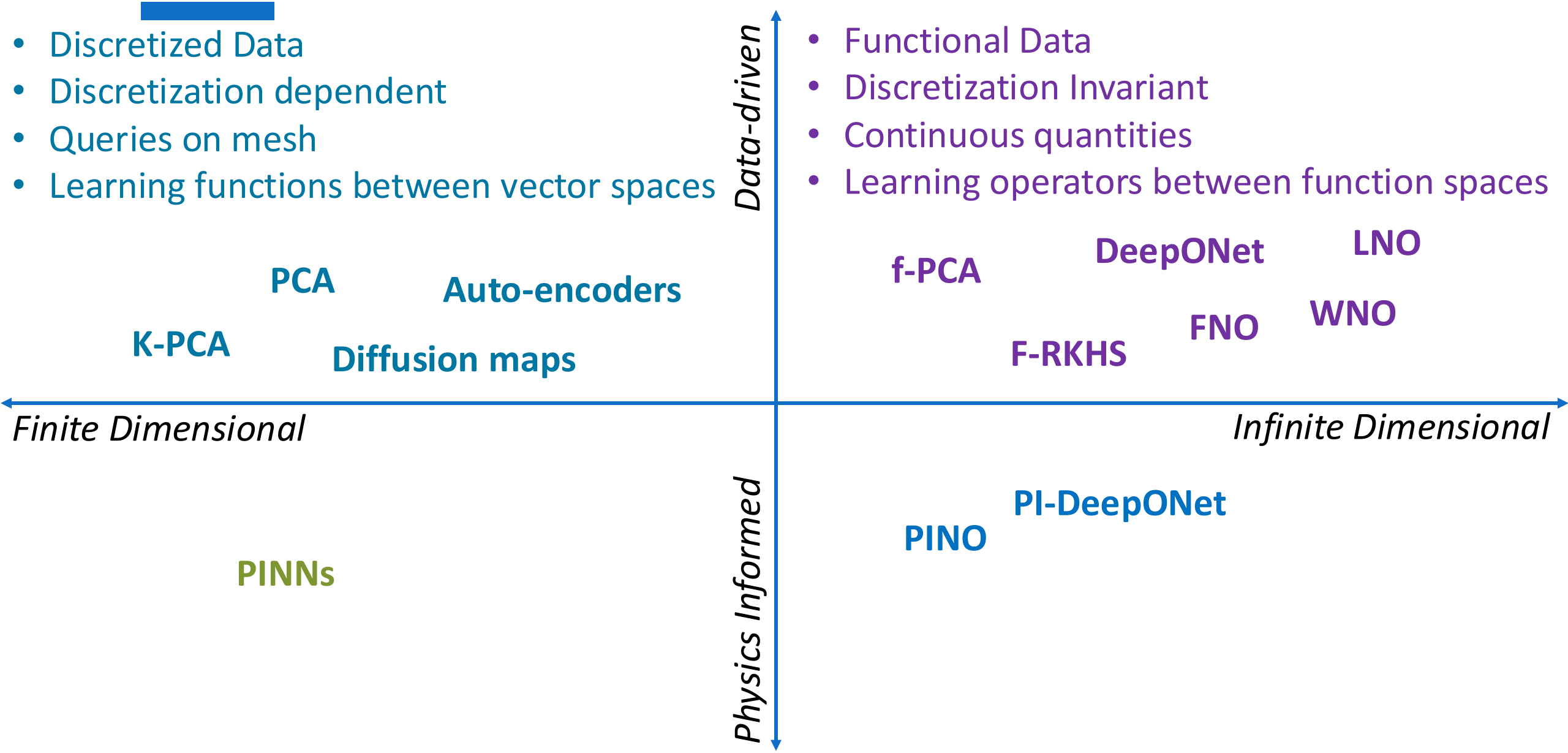
Challenges

- 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.

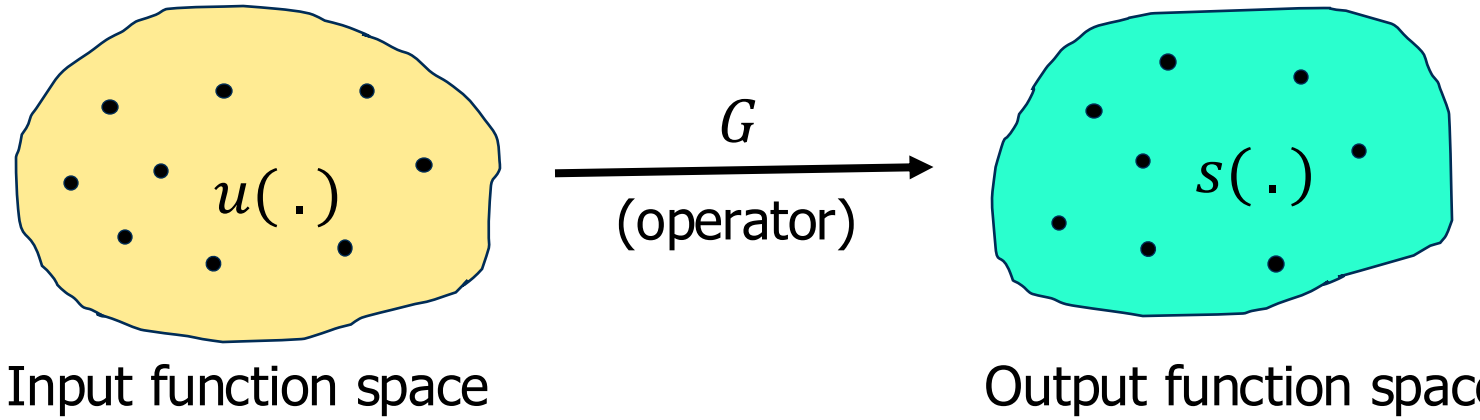
Few Surrogate Modeling Techniques

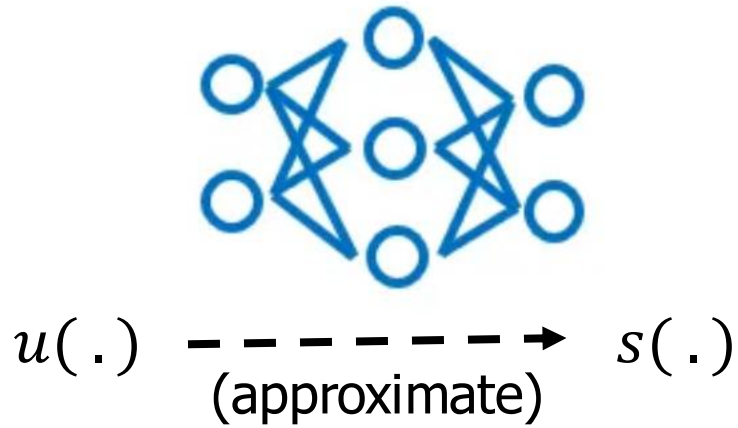
- Discretized Data
- Discretization dependent
- Queries on mesh
- Learning functions between vector spaces

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



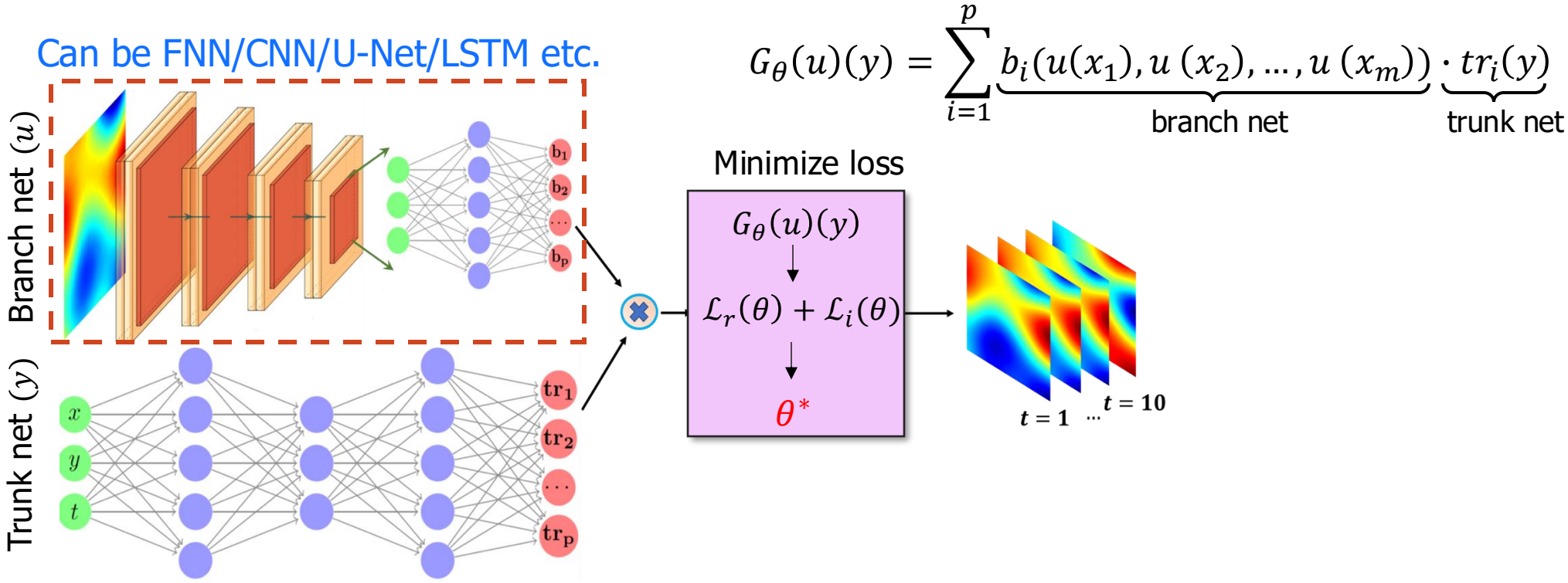
Operator Learning Framework





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

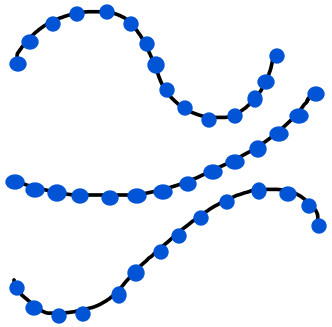


Training of DeepONet (Traditional)

Training Dataset

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

Consider a batch size (bs) = 3



[bs,m]

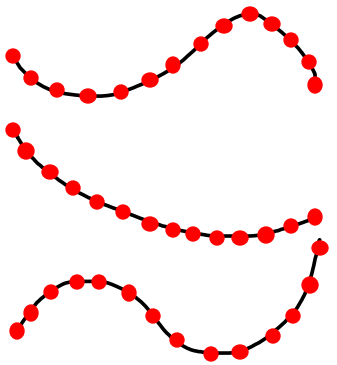
[bs,p]

Branch Net

$\mathcal{O}(bs)$

Input field 'm' sensor locations

$\otimes$ Einstein Summation [bs,n]



Output field predictions

$\mathcal{O}(bs * n)$

$x_1\ x_2 \ldots\ldots\ldots x_n$

'n' evaluation points

Trunk Net

[bs,n,1]

[bs,n,p]

Shortcomings

The training cost increases drastically as the number of
- training samples increases (larger batch size)
- evaluation points (n) increases

Therefore, not an efficient training architecture for high-dimensional systems.

The advantages of employing a stochastic gradient descent optimizer is not leveraged.

Our Proposed framework

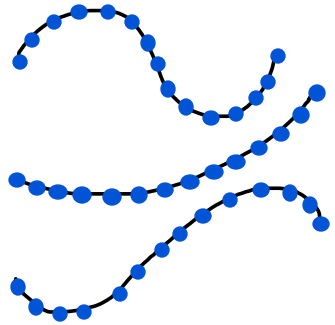
Efficient Training of Deep Neural Operator Networks via Randomized Sampling



Karumuri, Sharmila, Lori Graham-Brady, and Somdatta Goswami. "Efficient Training of Deep Neural Operator Networks via Randomized Sampling." *arXiv preprint arXiv:2409.13280* (2024).

Efficient Training of Deep Neural Operator Networks via Randomized Sampling

Consider a batch size (bs) = 3

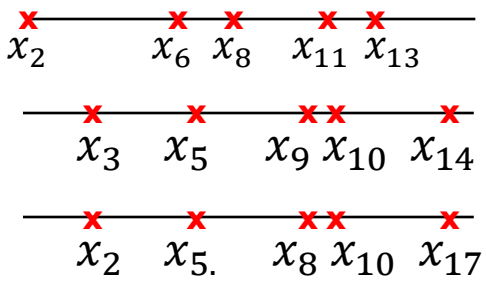


Input field 'm' sensor locations

[bs,m] [bs,p]

Branch Net

$\mathcal{O}(bs)$



'n_{eval}' evaluation points

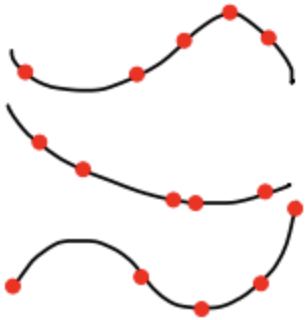
$\mathcal{O}(bs * n_{eval})$

Trunk Net

[bs, n_{eval},1] [bs, n_{eval},p]

Einstein Summation

[bs$* n_{eval}$]



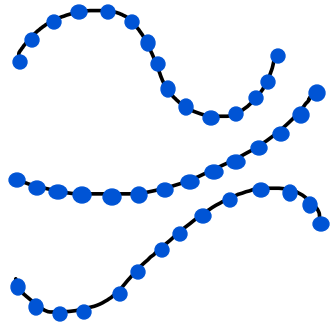
Output field predictions

Code in PyTorch

Idea: Randomized sampling of evaluation points during training

Efficient Training of Deep Neural Operator Networks via Randomized Sampling

Consider a batch size (bs) = 3



[bs,m]

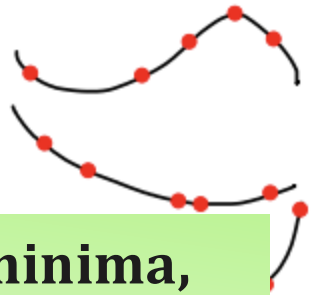
Input field 'm' sensor locations

Branch Net

$\mathcal{O}(bs)$

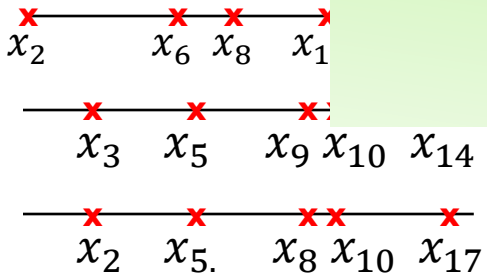
[bs,p]

Einstein Summation



Output field predictions

The lower effective batch size leads to flatter minima, avoiding sharp minima and results in better generalization.



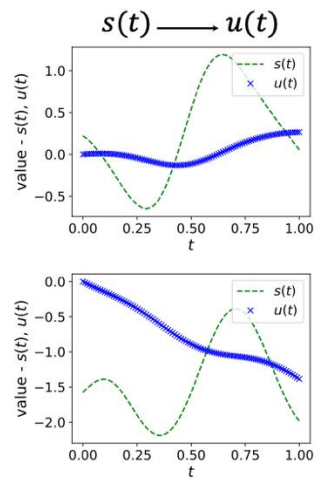
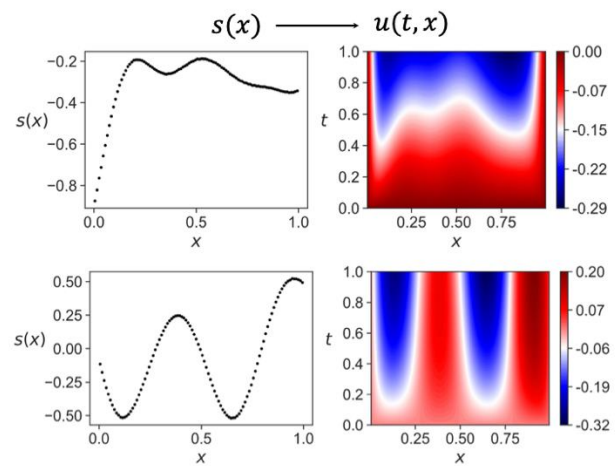
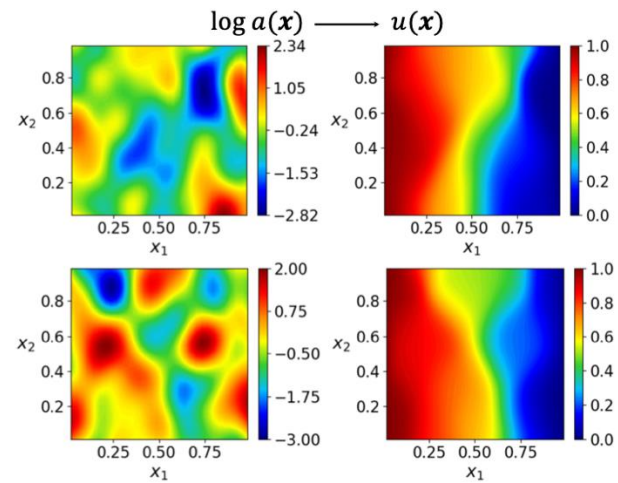
'n_{eval}' evaluation points

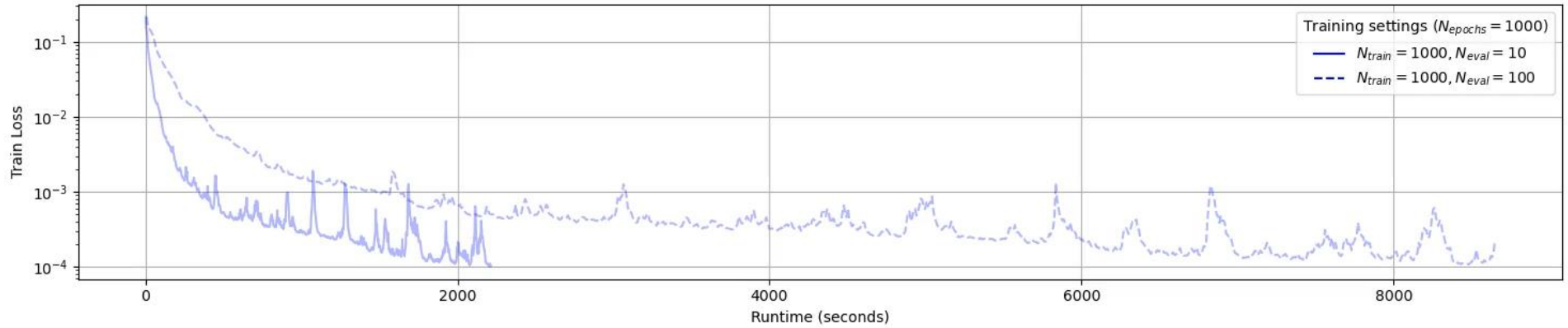
Net

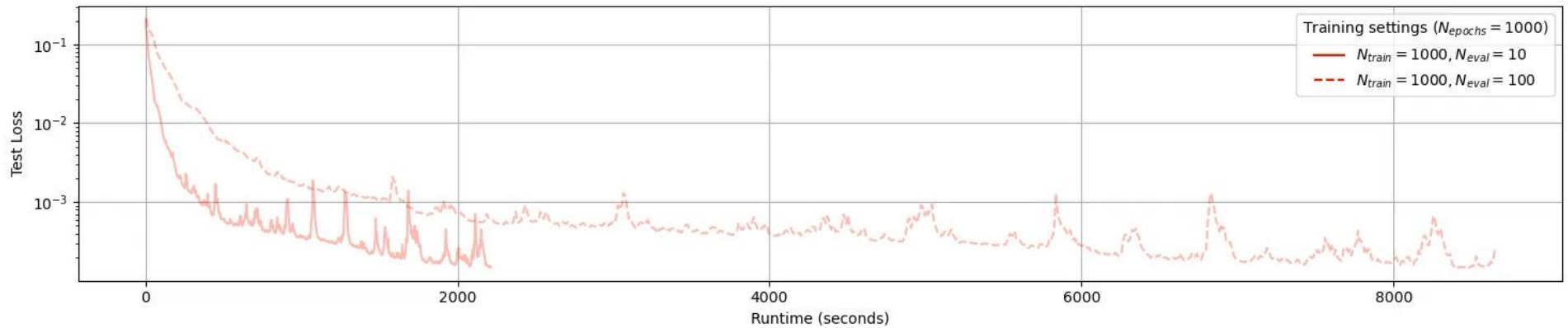
[bs, n_{eval},1] [bs, n_{eval},p]

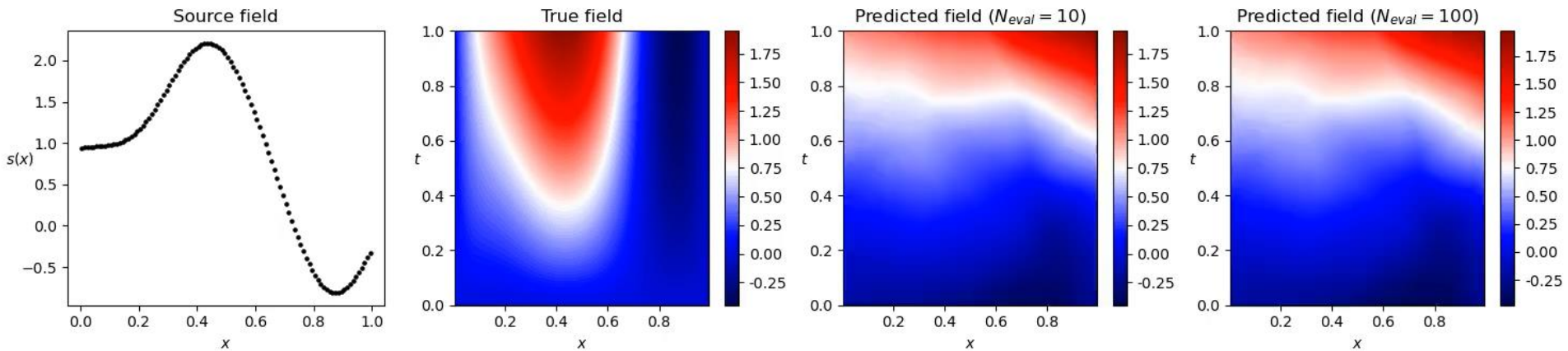
Code in PyTorch

Idea: Randomized sampling of evaluation points during training

Case	Dynamical System	Diffusion-reaction	Heat Equation
Model Output	$\frac{du}{dt} = s(t),$ $u(0) = 0$ and $t \in [0, 1]$ $\mathcal{G}_{\boldsymbol{\theta}} : s(t) \to u(t).$	$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + ku^2 + s(x),$ $D = 0.01,\ k = 0.01,$ $(t, x) \in (0, 1] \times (0, 1],$ $u(0, x) = 0,\ x \in (0, 1)$ $u(t, 0) = 0,\ t \in (0, 1)$ $u(t, 1) = 0,\ t \in (0, 1)$ $\mathcal{G}_{\boldsymbol{\theta}} : s(x) \to u(t, x).$	$-\nabla \cdot (a(\boldsymbol{x})\nabla u(\boldsymbol{x})) = 0,$ $\boldsymbol{x} = (x_1, x_2),$ $\boldsymbol{x} \in \Omega = [0, 1]^2,$ $u(0, x_2) = 1,\ u(1, x_2) = 0,$ $\frac{\partial u(x_1, 0)}{\partial n} = \frac{\partial u(x_1, 1)}{\partial n} = 0,$ $\mathcal{G}_{\boldsymbol{\theta}} : a(\boldsymbol{x}) \to u(\boldsymbol{x}).$
Input Function	$s(t) \sim \mathrm{GP}(0, k(t, t')),$ $\ell_t = 0.2,\ \sigma^2 = 1.0,\ t \in [0, 1]$ $k(t, t') = \sigma^2 \exp\left\{-\frac{\|t - t'\|^2}{2\ell_t^2}\right\}.$	$s(x) \sim \mathrm{GP}(0, k(x, x')),$ $\ell_x = 0.2,\ \sigma^2 = 1.0,$ $k(x, x') = \sigma^2 \exp\left\{-\frac{\|x - x'\|^2}{2\ell_x^2}\right\}.$	$\log(a(\boldsymbol{x})) \sim \mathrm{GP}(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))$ $\mu(\boldsymbol{x}) = 0,\ \ell_{x_1} = 0.1,\ \ell_{x_2} = 0.15,\ \sigma^2 = 1.0$ $k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left\{-\sum_{i=1}^{2} \frac{\|x_i - x_i'\|^2}{2\ell_{x_i}^2}\right\}.$
Samples			

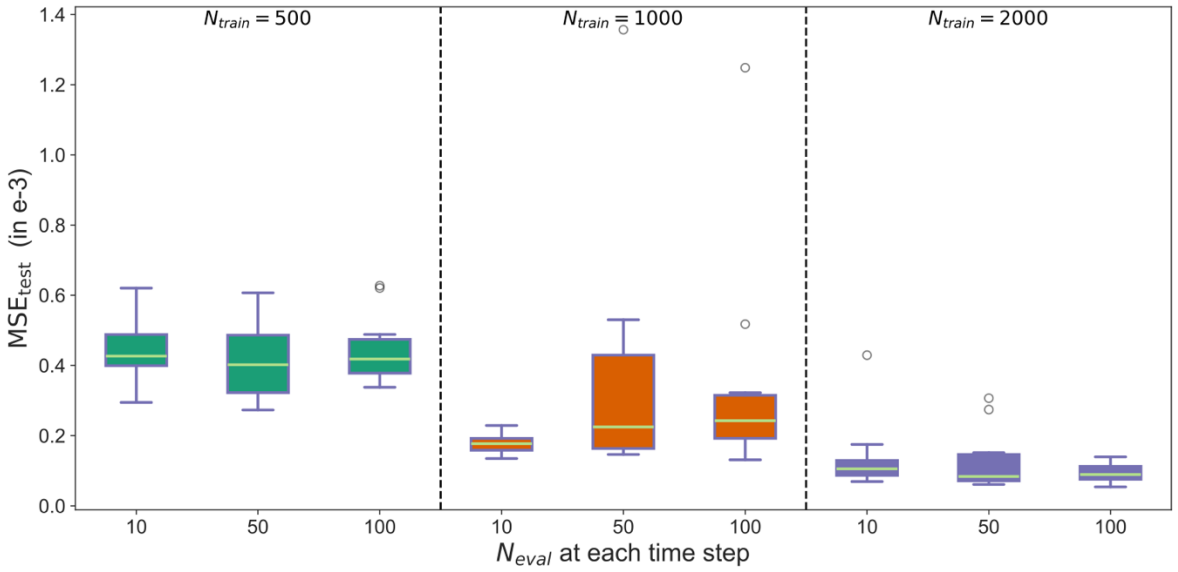

Training settings ($N_{epochs} = 1000$)
$N_{train} = 1000, N_{eval} = 10$
$N_{train} = 1000, N_{eval} = 100$
Train Loss
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
2000
4000
6000
8000
Runtime (seconds)


Training settings ($N_{epochs} = 1000$)
$N_{train} = 1000, N_{eval} = 10$
$N_{train} = 1000, N_{eval} = 100$
Test Loss
10^{-1}
10^{-2}
10^{-3}
0
2000
4000
6000
8000
Runtime (seconds)

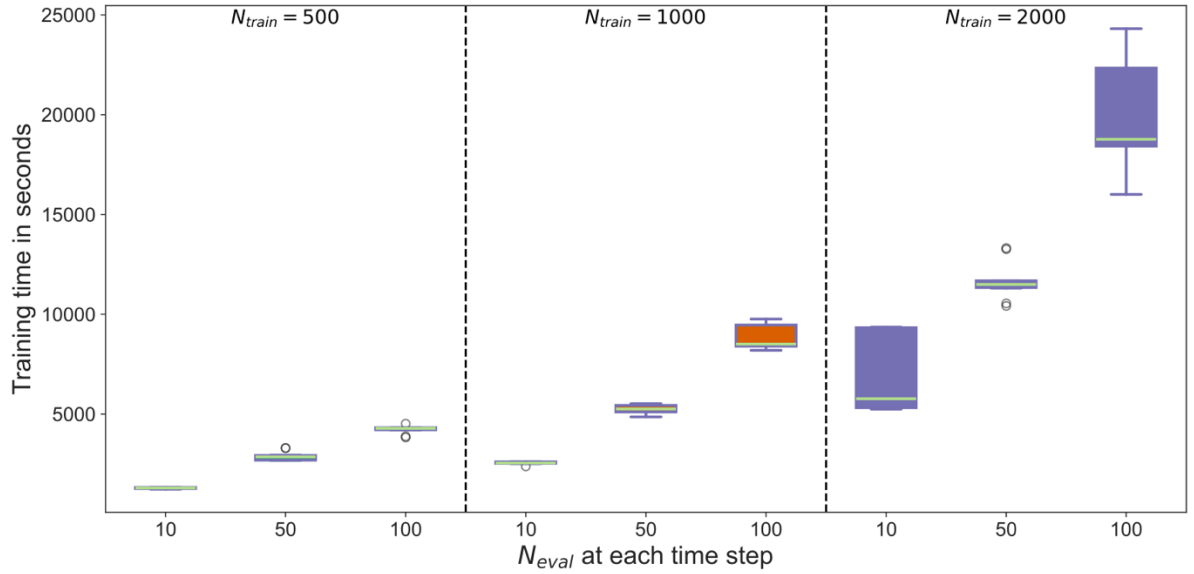

Source field
$s(x)$
2.0
1.5
1.0
0.5
0.0
−0.5
0.0
0.2
0.4
0.6
0.8
1.0
x
True field
Predicted field ($N_{eval} = 10$)
Predicted field ($N_{eval} = 100$)
t
0.0
0.2
0.4
0.6
0.8
1.0
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
-0.25

Parametric Study

Mean Square Error



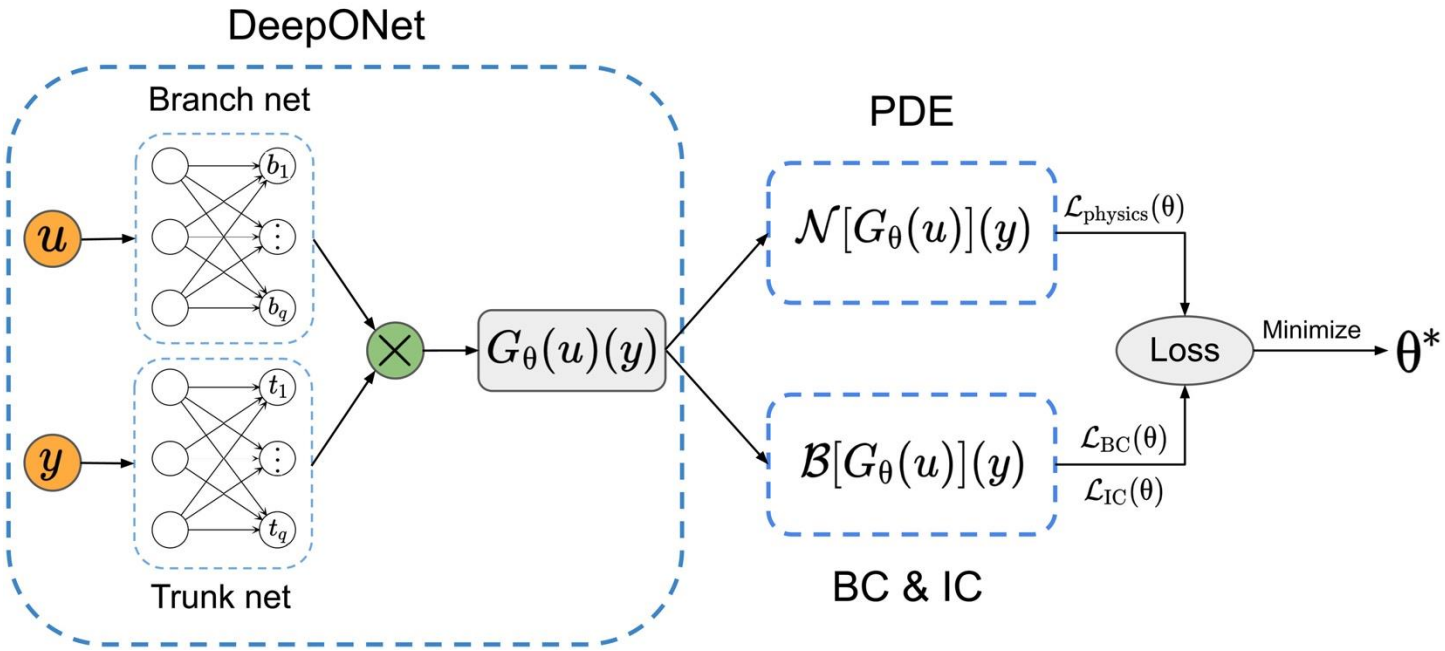
Training Time



Conclusion: Our approaches achieves same accuracy in drastically lower training time.

Physics-Informed DeepONet

Addressing the data-hungry behavior of DeepONet



- Wang et.al "Learning the solution operator of parametric partial differential equations with physics-informed DeepONets" Sceince Advances, 2021
- Goswami et al. "A physics-informed variational DeepONet for brittle fracture." *CMAME, 2022*.

Shortcomings

↑

Training is extremely expensive. So, never made it to common practice.

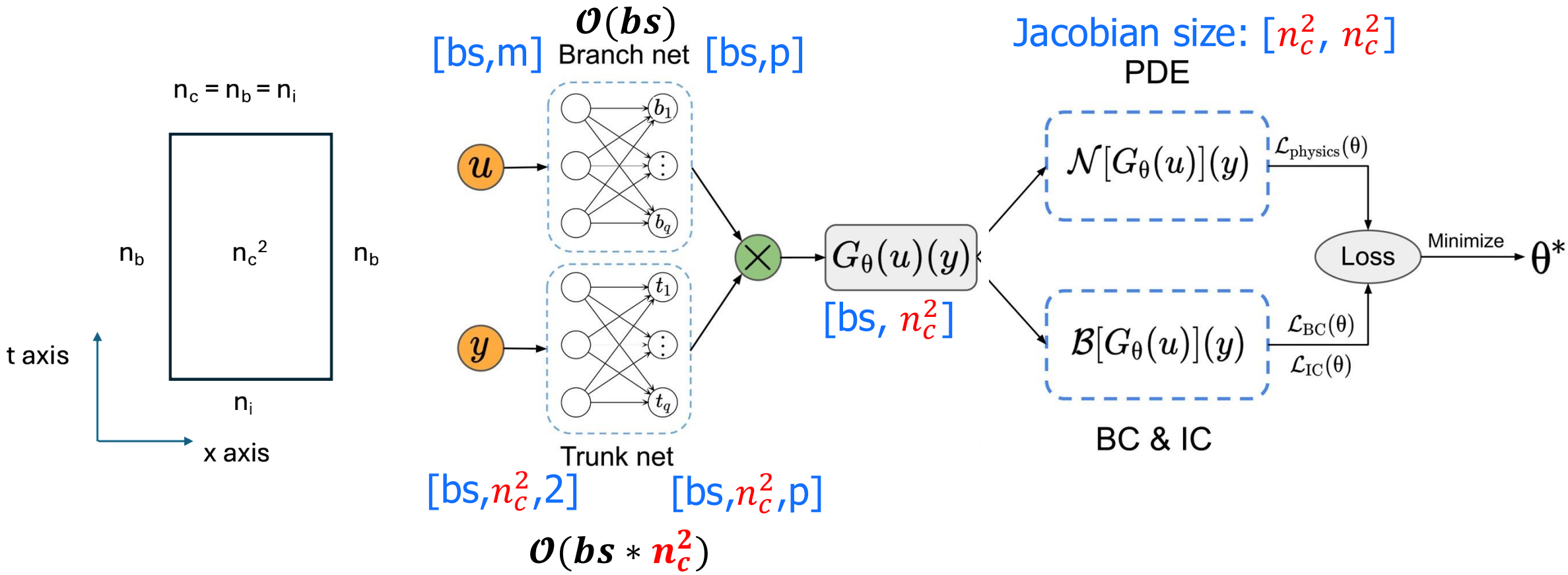
Our Proposed framework

Separable DeepONet: Breaking the Curse of Dimensionality in Physics-Informed Machine Learning

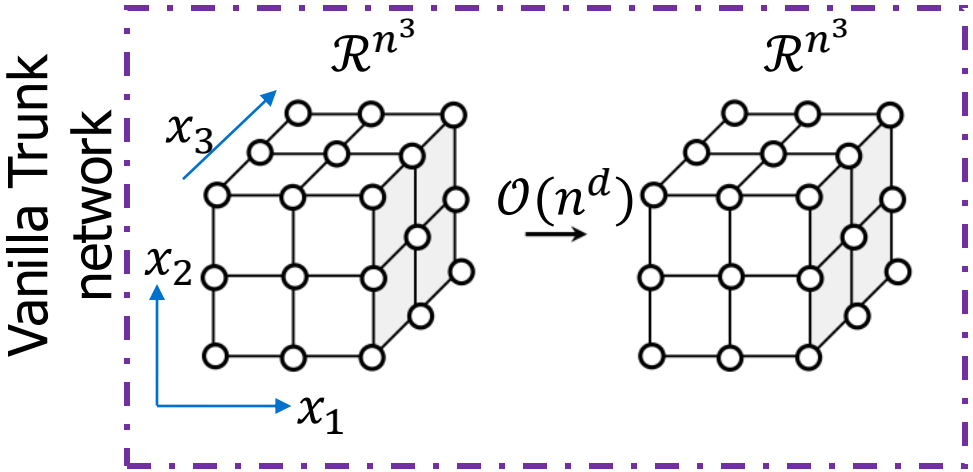


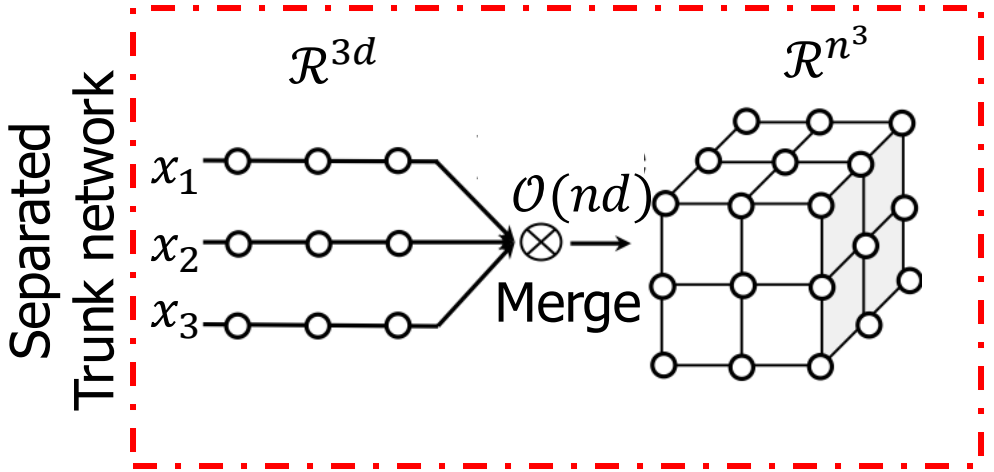

Mandl, Luis, Somdatta Goswami, Lena Lambers, and Tim Ricken. "Separable DeepONet: Breaking the Curse of Dimensionality in Physics-Informed Machine Learning." *arXiv preprint arXiv:2407.15887* (2024).

Vanilla – Physics Informed DeepONet



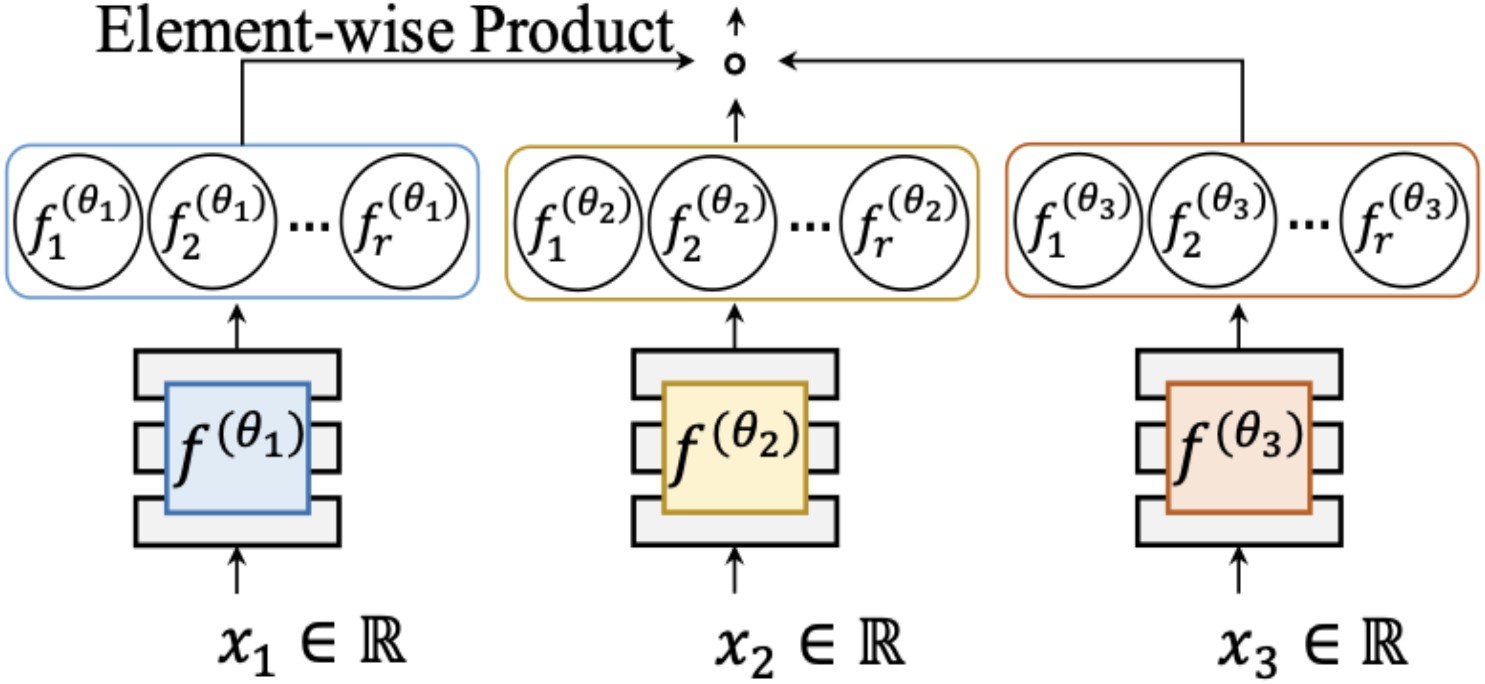
Introducing Separation of Variables





Modifying the trunk network

$$\hat{u}(x) = \sum_{j=1}^{r} f_j^{(\theta_1)}(x_1) f_j^{(\theta_2)}(x_2) f_j^{(\theta_3)}(x_3)$$



Separable - Physics Informed DeepONet

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

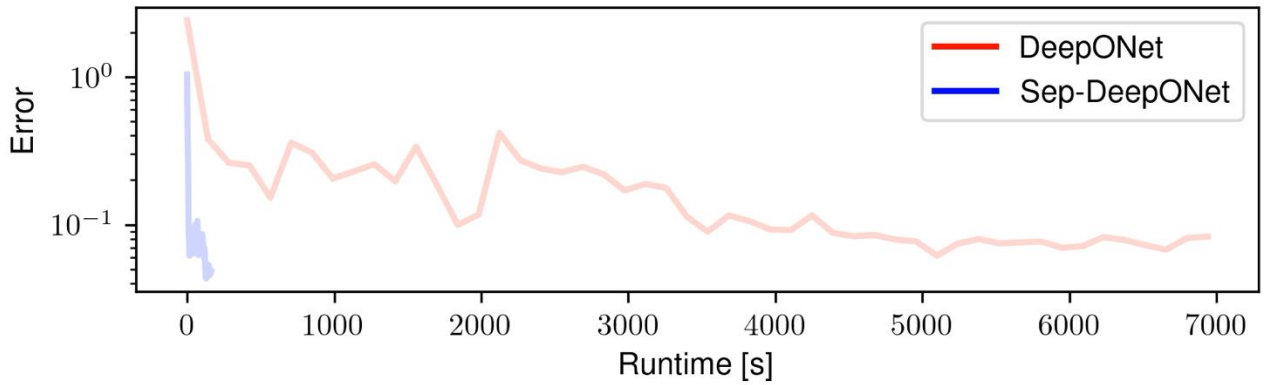
Burgers' Equation

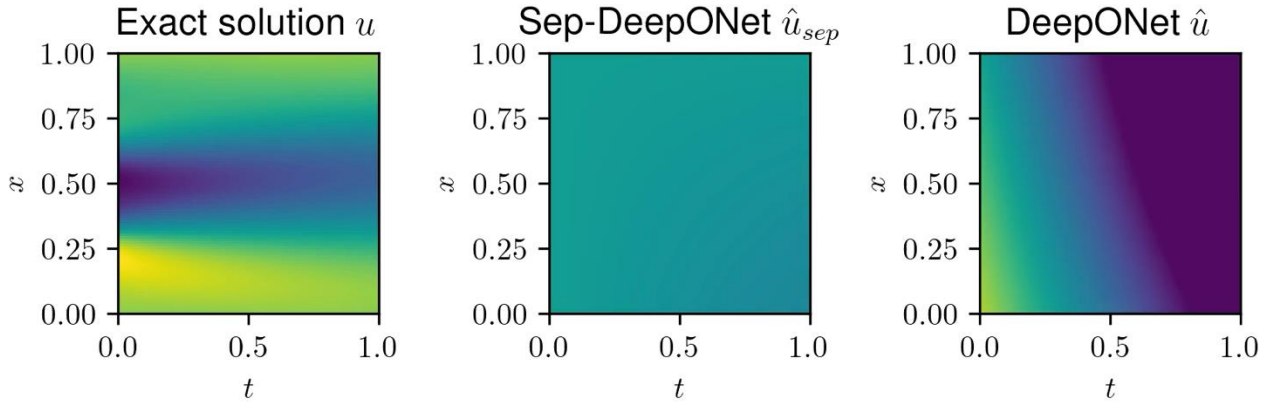
$$\frac{\partial s(x,t)}{\partial t} + s\frac{\partial s(x,t)}{\partial x} - \nu\frac{\partial^2 s(x,t)}{\partial x^2} = 0,$$

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

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

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





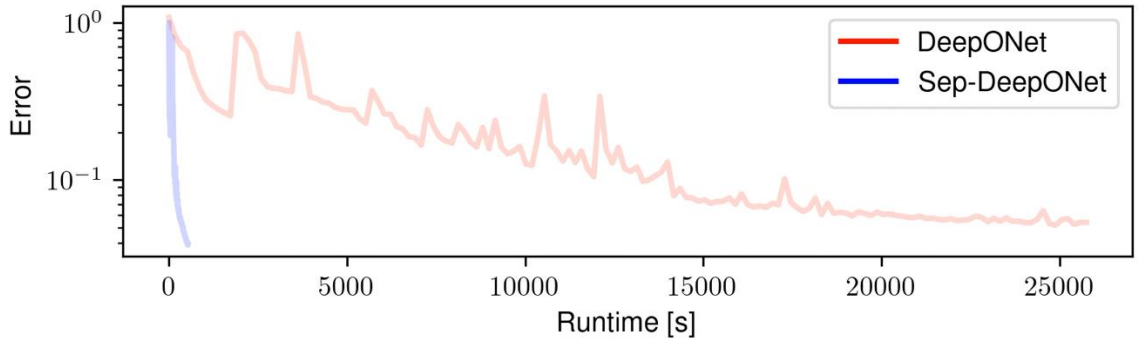
Model	Branch	Trunk	p	r	Parameters	$\mathcal{L}_2$ rel. err.	Runtime [s]	Runtime improvment
Vanilla PI-DeepONet	6×[100]	6×[100]	100	-	131,701	5.14e-2	6,829.2	-
Sep-PI-DeepONet	6×[100]	6×[100]	50	50	672,151	6.24e-2	182.1	97,33%
	6×[100]	6×[100]	20	20	244,921	6.04e-2	197.8	97,10%
	6×[100]	6×[50]	20	20	129,221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

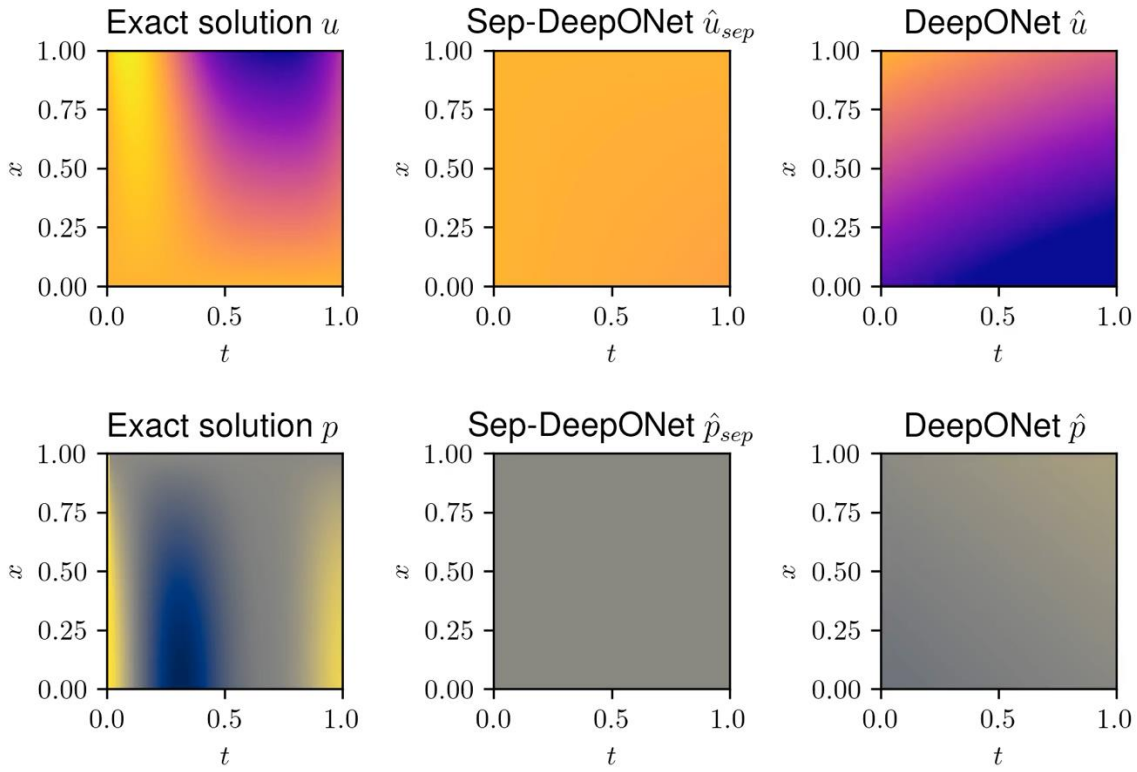
Biot's Consolidation

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

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

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





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

Our Proposed framework

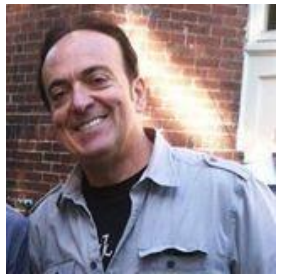
nature communications

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

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

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

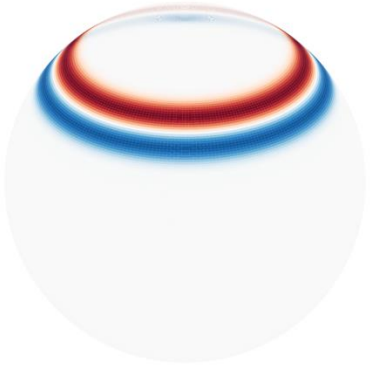
$$h'(\lambda,\phi) = \hat{h}\cos(\phi)\mathrm{e}^{-(\lambda/\alpha)^2}\mathrm{e}^{-[(\phi_2-\phi)/\beta]^2}$$

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

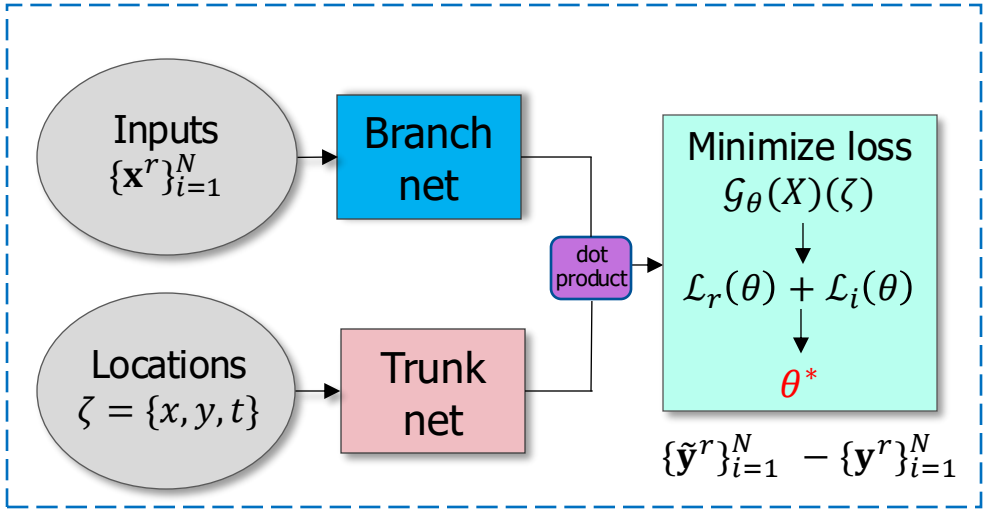
Atmospheric Flow

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

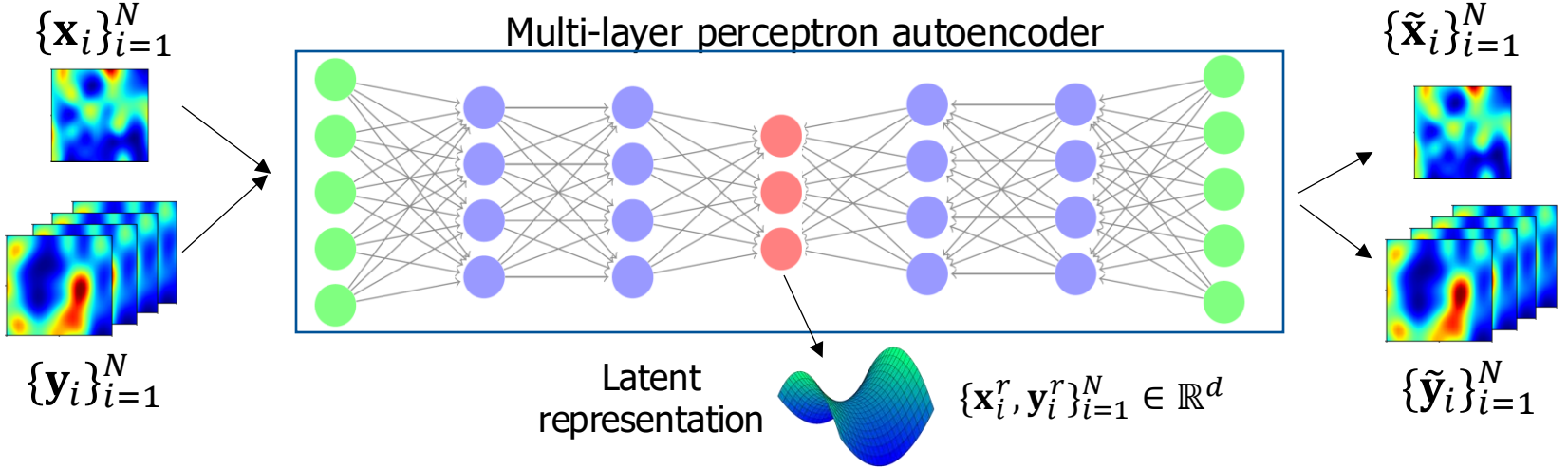


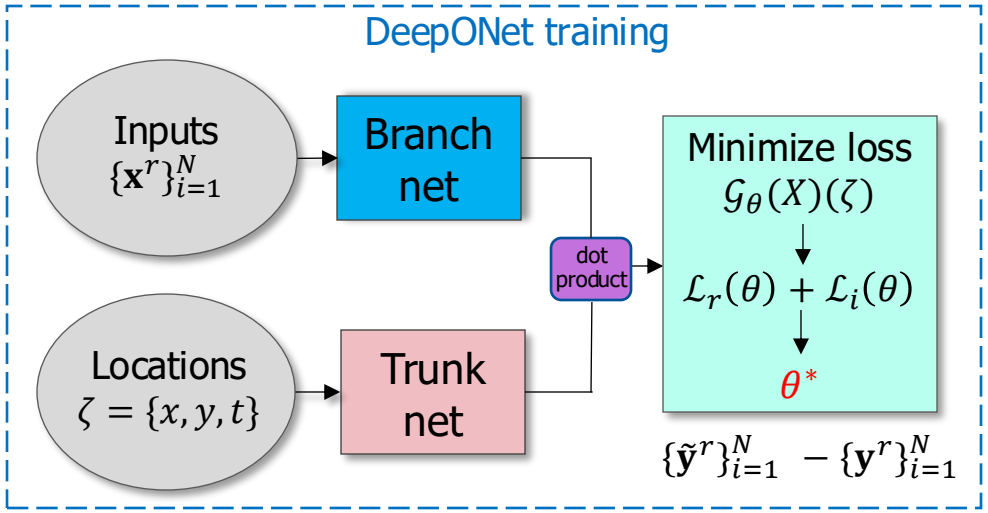
Latent DeepONet for time-dependent PDEs

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

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

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



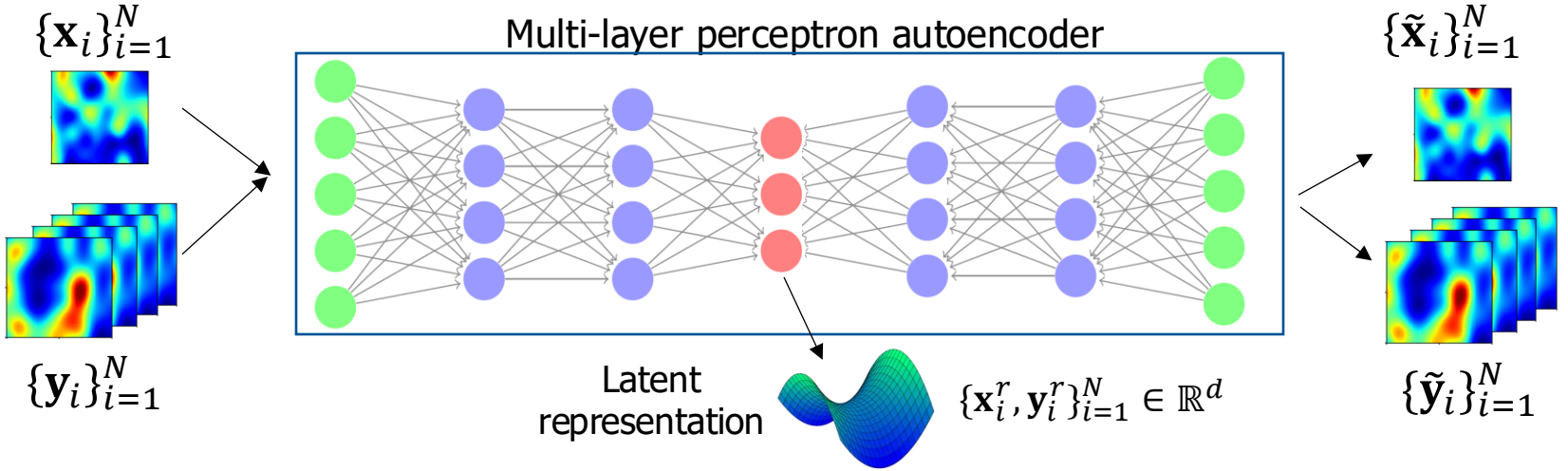


Latent DeepONet for time-dependent PDEs

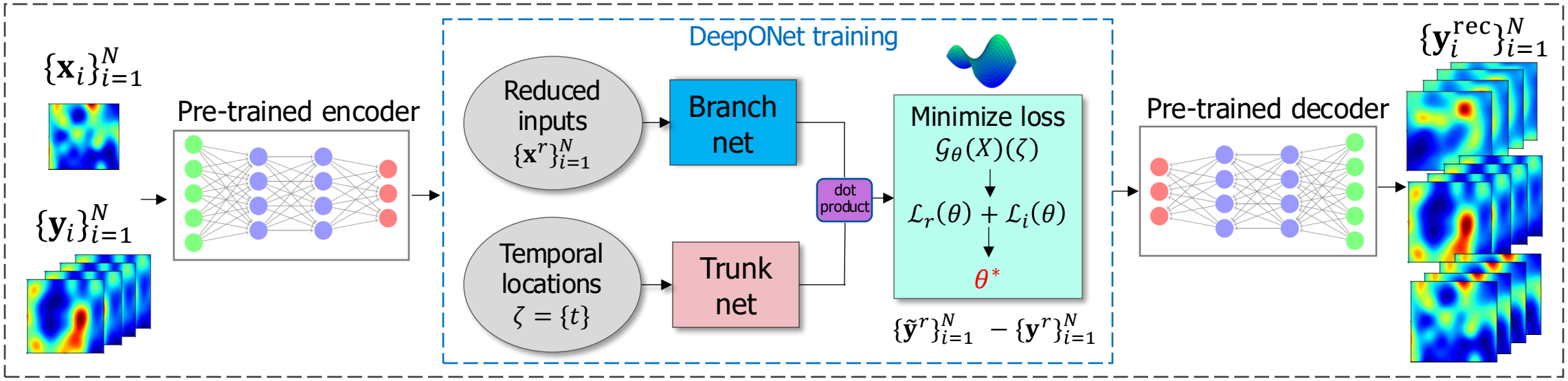
Operator $\mathcal{G}: \mathcal{X} \to \mathcal{Y}$

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

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

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

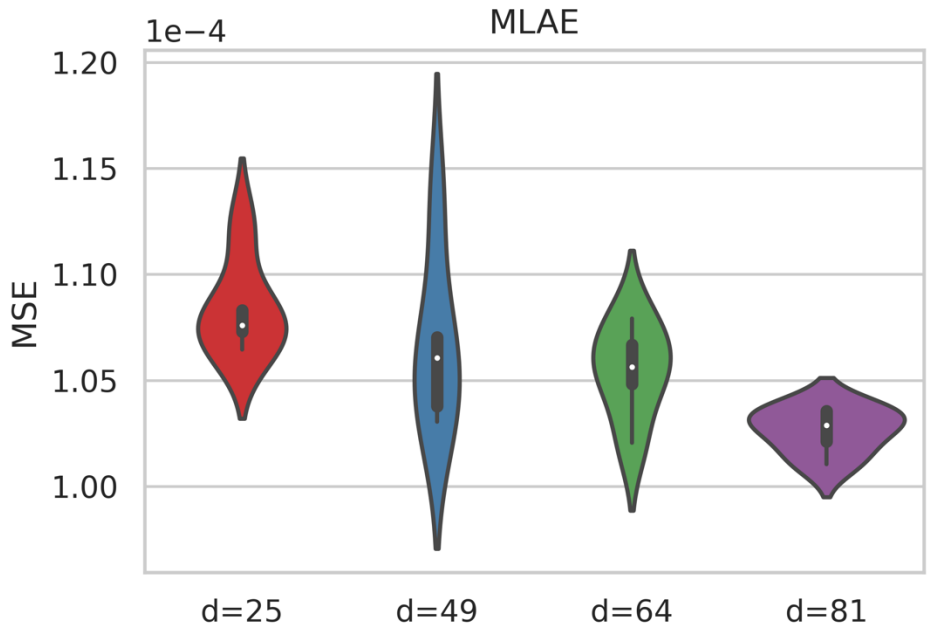
Results

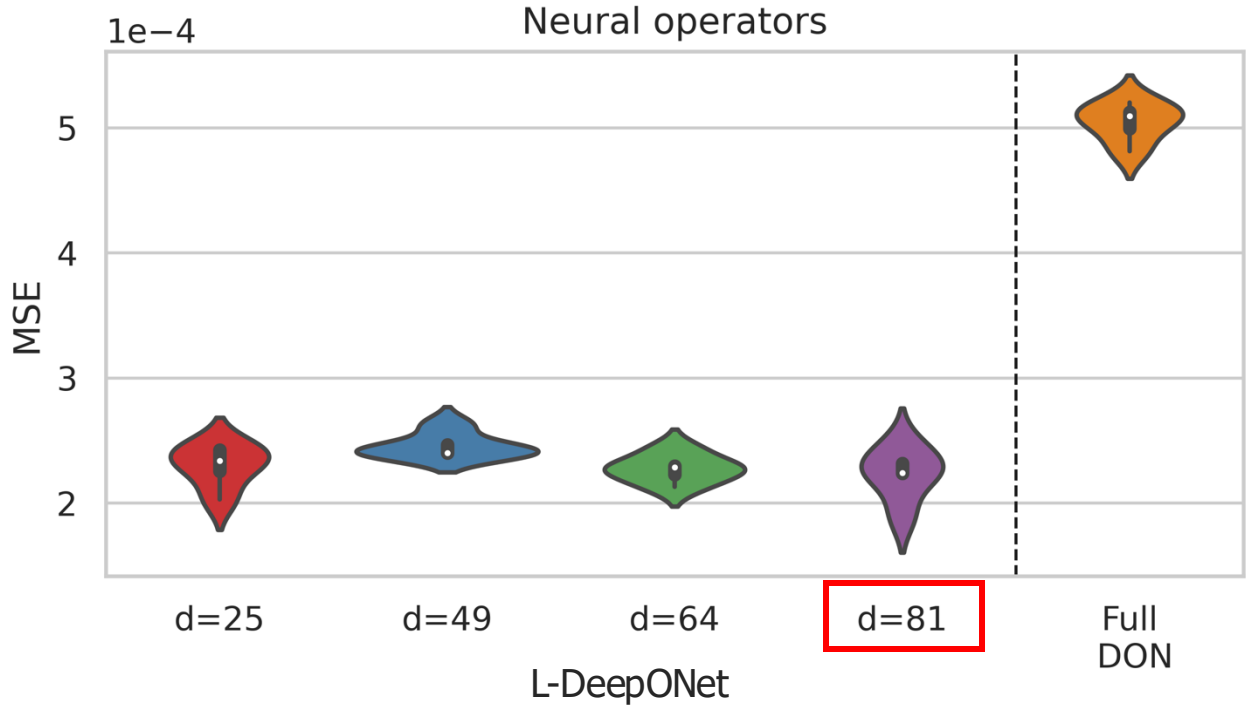
- $\Omega = [0,2\pi]x[0,2\pi]$, $(n_x \, x \, n_y) = (256x256)$ mesh points
- Output dimensionality: $72x256x256 = 4{,}718{,}592$
- Simulation: $t = [0,360h], \delta t = 0.1\bar{6}h$, Time steps: $n_t = 72$

Training Time (seconds)

MLAE + Latent DON: 15, 218

Full DON: 379,022



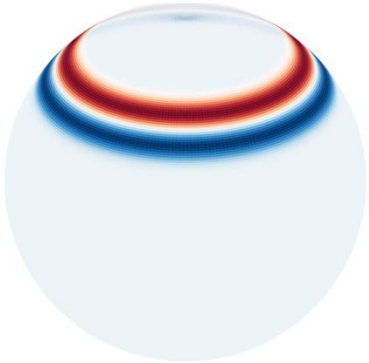


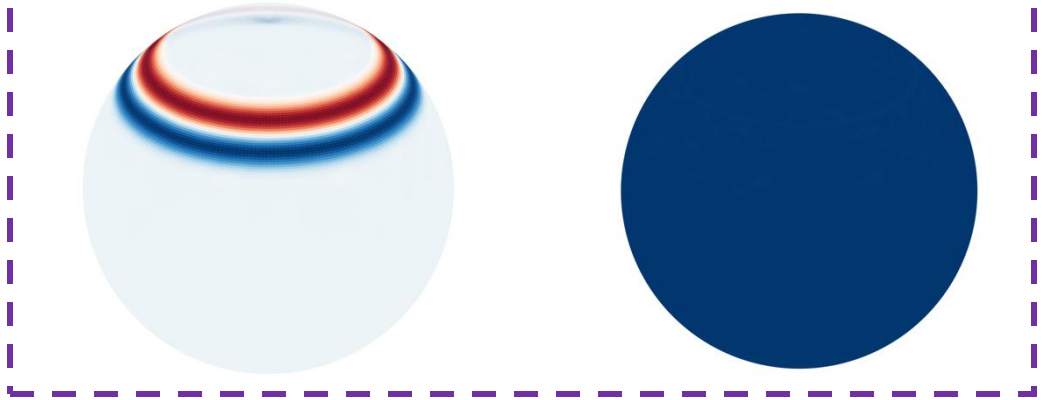
Results

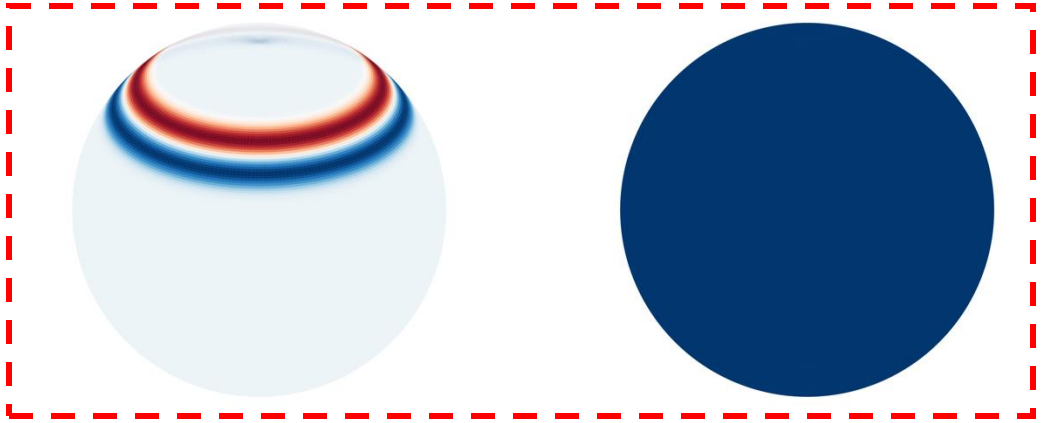
Prediction

Error

Reference

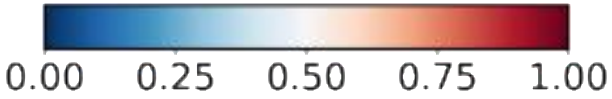




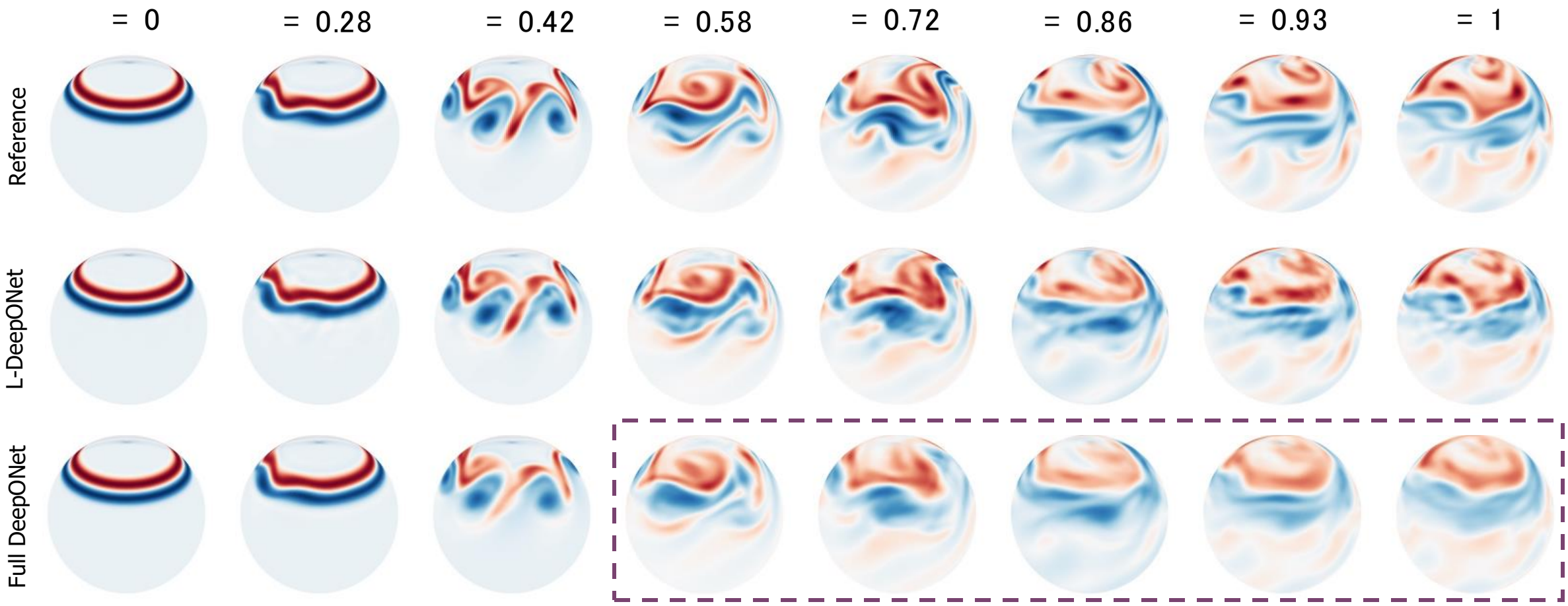


Latent DeepONet
(275,475 Parameters)

DeepONet
(327,872 Parameters)

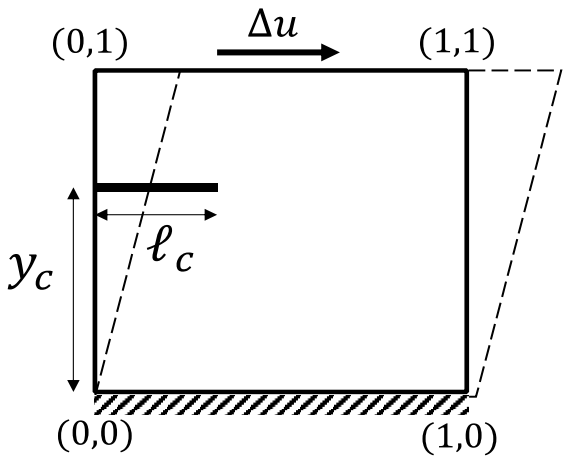


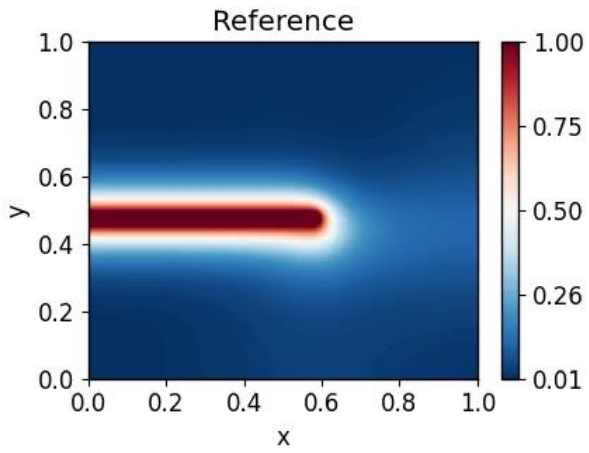
Latent DeepONet and Full DeepONet

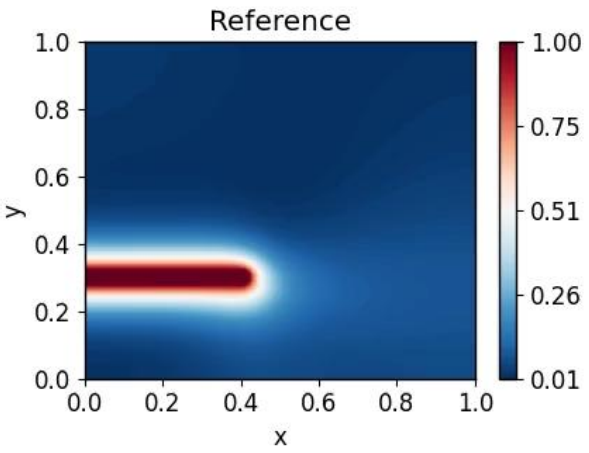


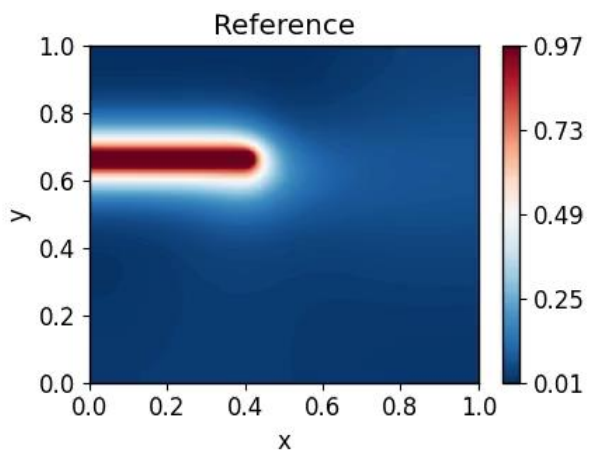
Fracture: Shear failure of plate with notch

- Unit square plate with horizontal crack
- Both location y_c and length ℓ_c of the crack are considered random
- Boundary conditions: $u(x,0) = v(x,0) = 0$, $u(x,1) = \Delta u$
- Data: $N = 261$, $y_c \in [0.2, 0.675]$, $\ell_c \in [0.3, 0.65]$
- Input dimension: $162\text{x}162$ Output dimension: $8\text{x}162\text{x}162$





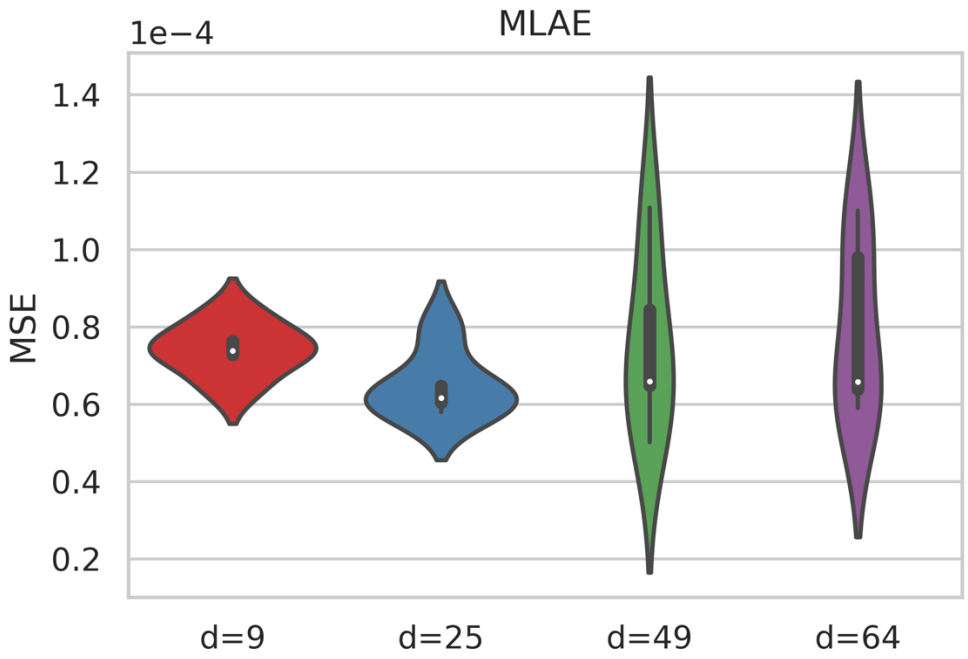


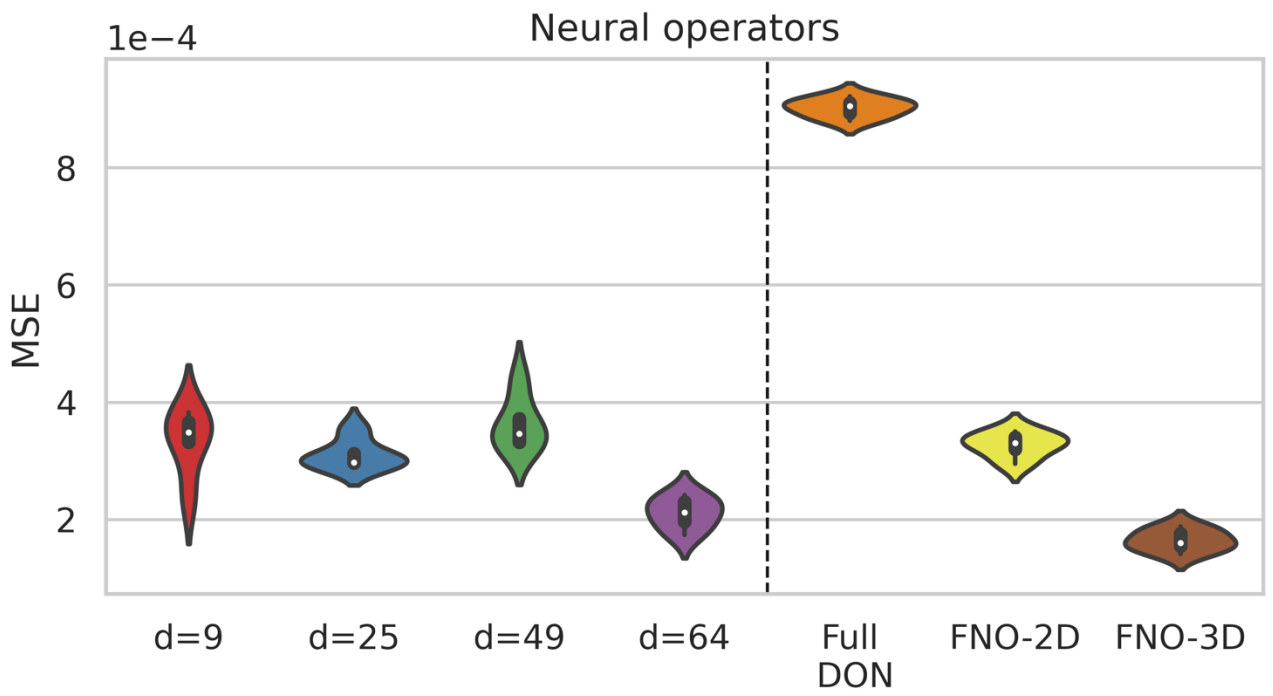


Goswami et al. (2021) A physics-informed variational DeepONet for predicting the crack path in brittle materials

Fracture: Shear failure of plate with notch

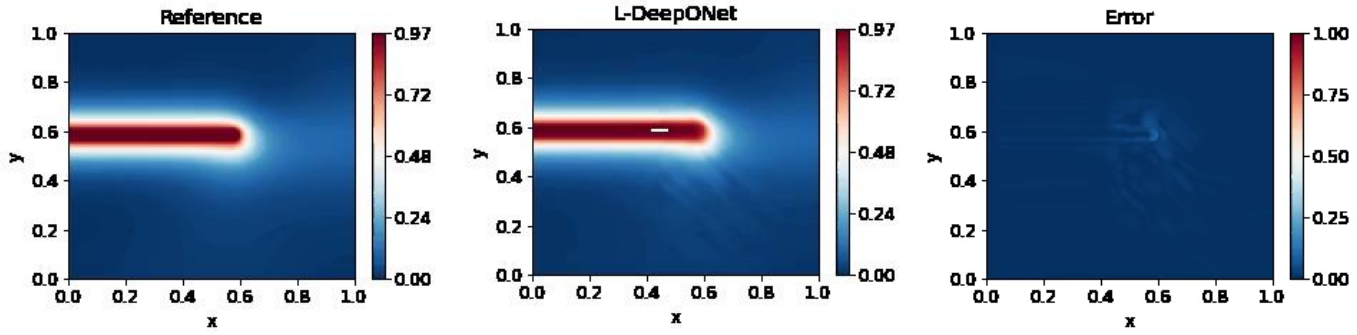
Error metric: $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$



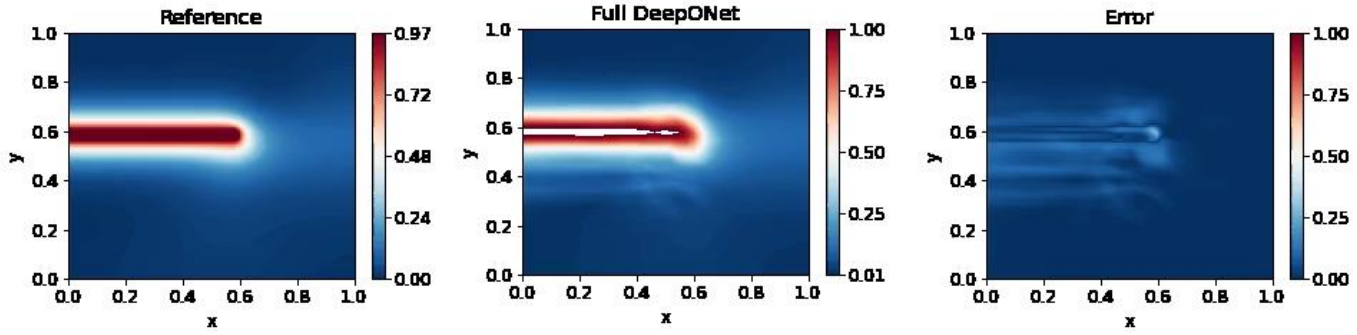


Comparison with Benchmark DeepONet

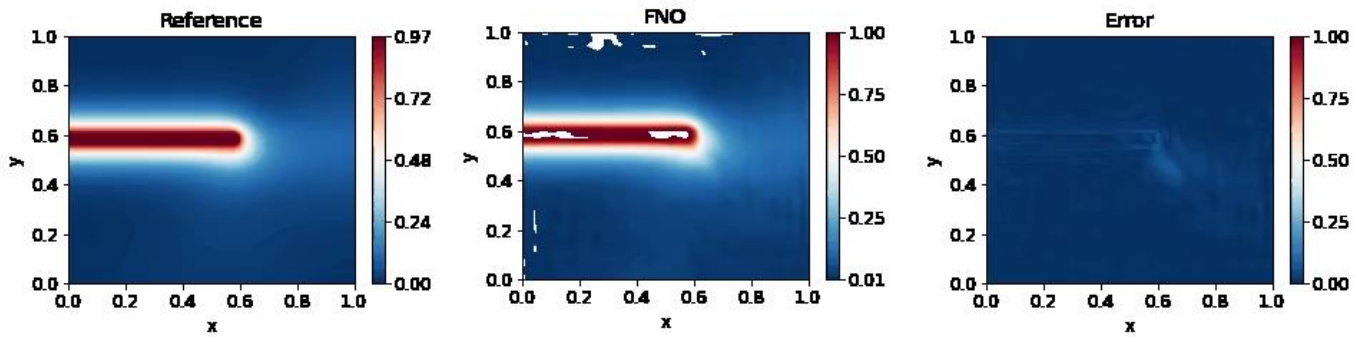
L-DeepONet



Full DeepONet



FNO



Shortcomings

The framework requires voluminious training data.

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

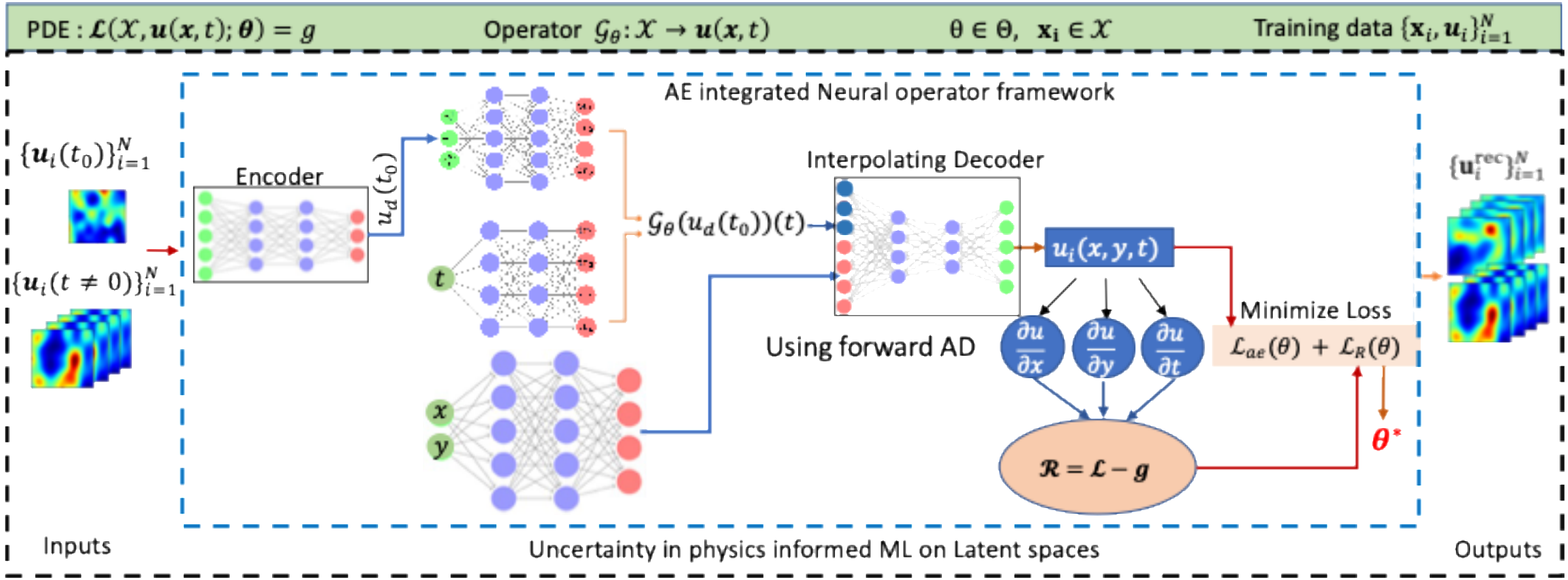
Our Proposed framework

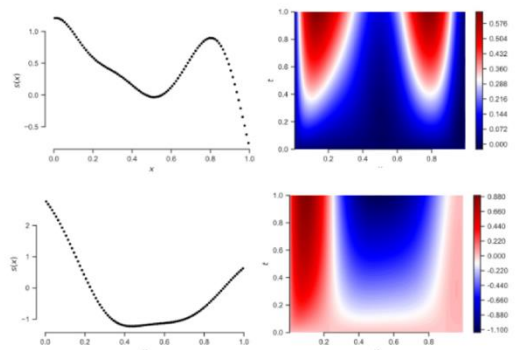
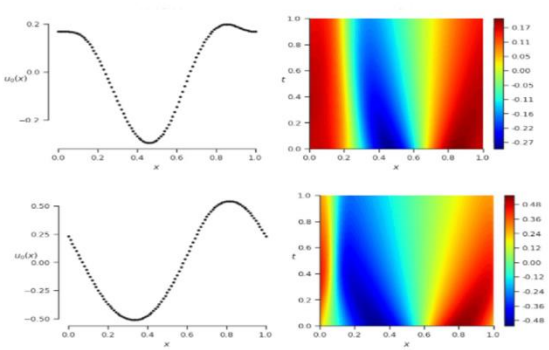
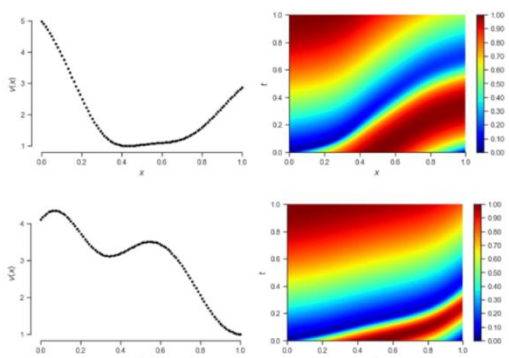
Physics-Informed Latent Neural Operator: Integrating Physics and Data using Reduced Order Modeling



Manuscript in preparation

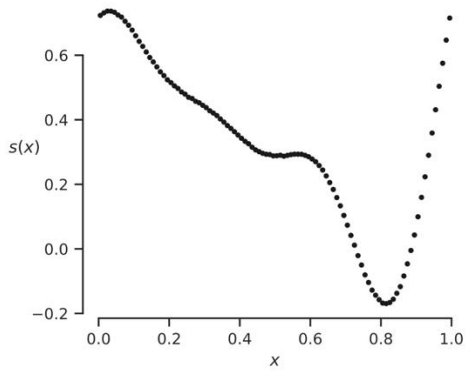
One-shot Learning: Neural operator and ROM



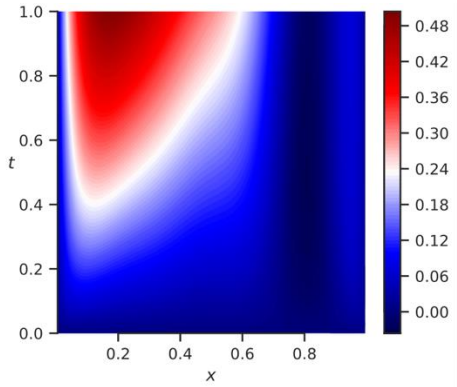
Case	Diffusion-reaction dynamics	Burgers' transport dynamics	Advection
PDE	$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} + ku^2 + s(x),$ $D = 0.01,\ k = 0.01,$ $(t, x) \in (0, 1] \times (0, 1],$ $u(0, x) = 0,\ x \in (0, 1)$ $u(t, 0) = 0,\ t \in (0, 1)$ $u(t, 1) = 0,\ t \in (0, 1)$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

Reaction Diffusion Dynamics

Source
(Input function)



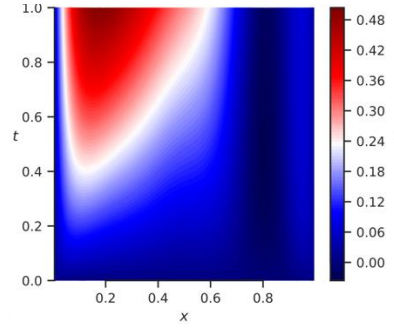
Solution
(Ground Truth)

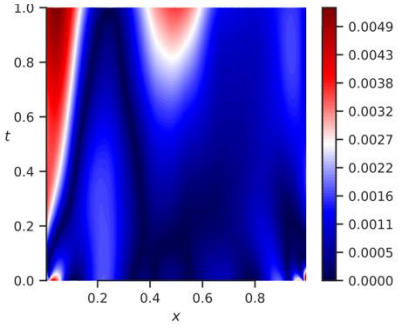


Prediction

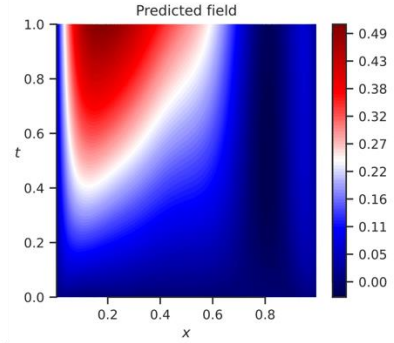
Error

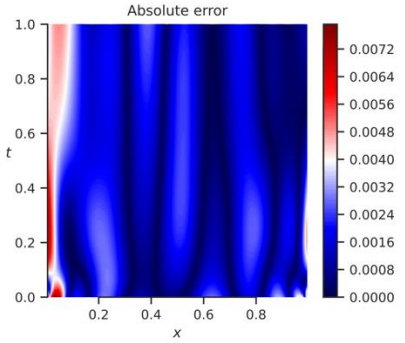
PI-Vanilla-NO



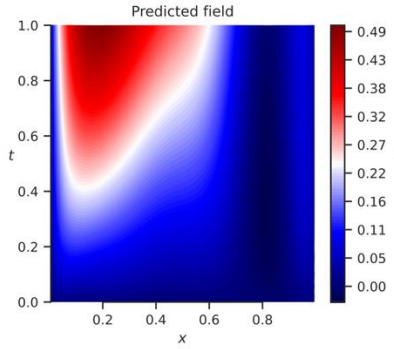


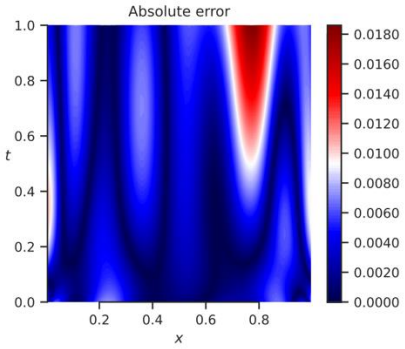
PI-Latent-NO (with PCA)



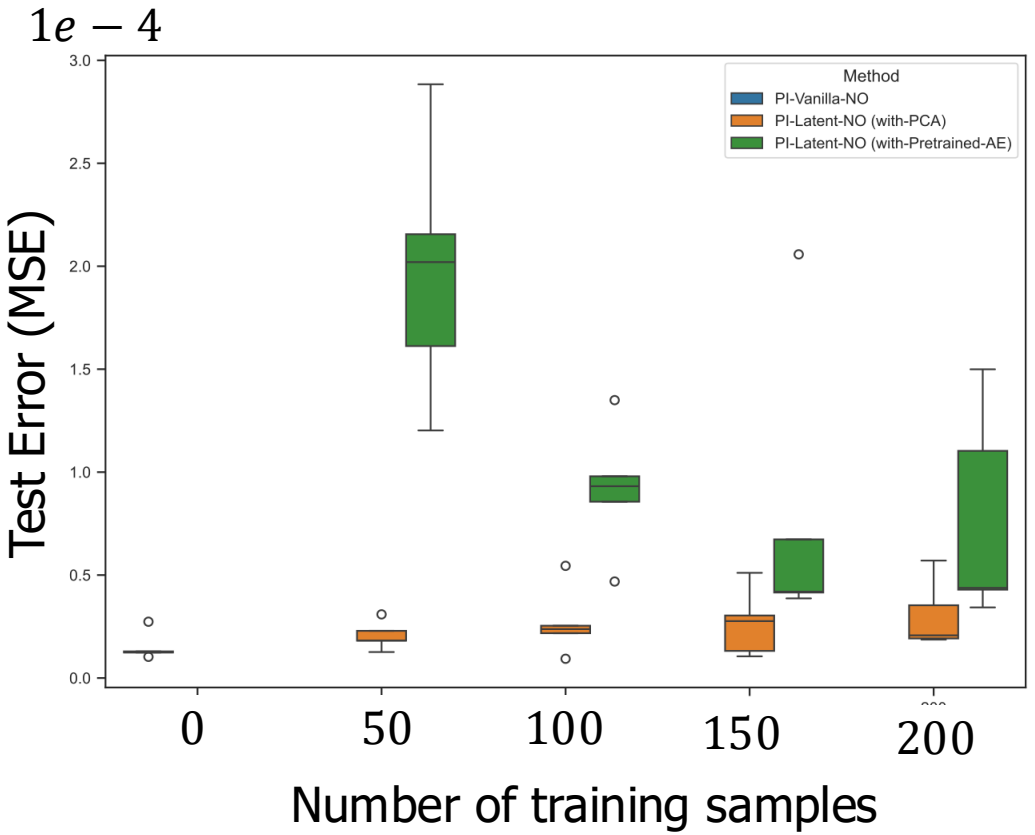


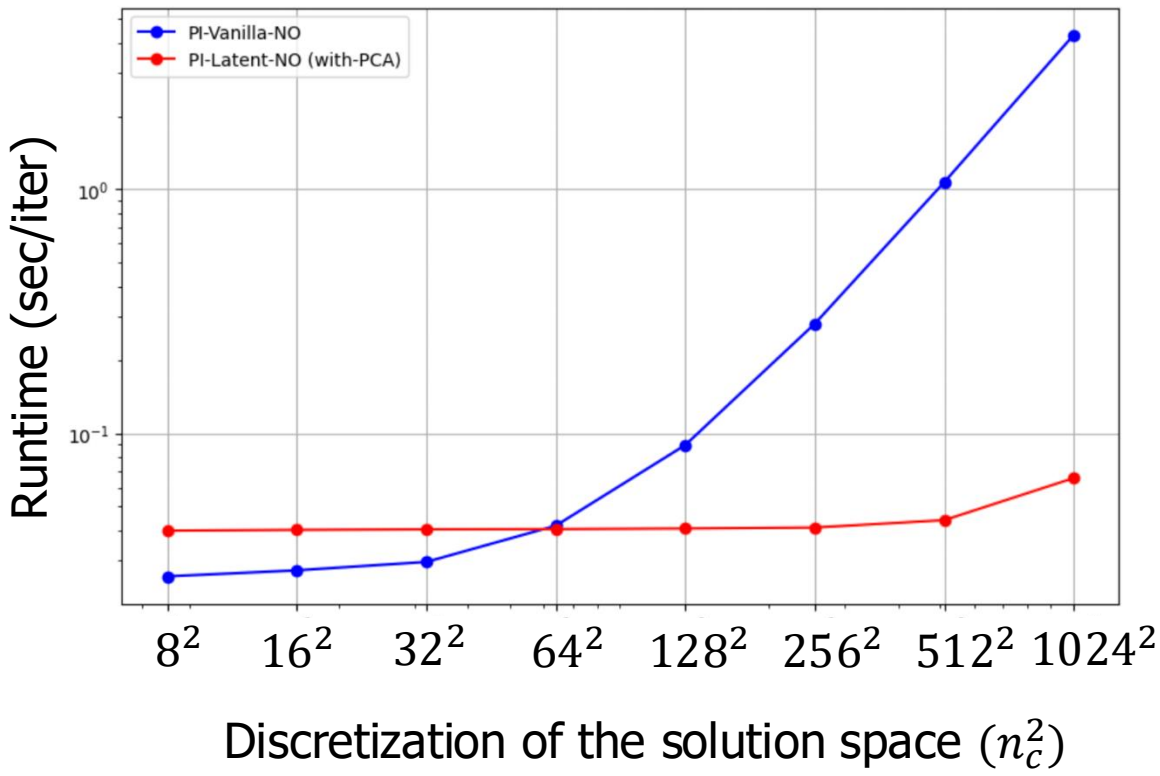
PI-Latent-NO (with AE)





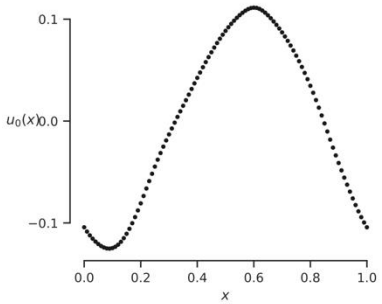
Scaling Test on the Reaction Diffusion Dynamics



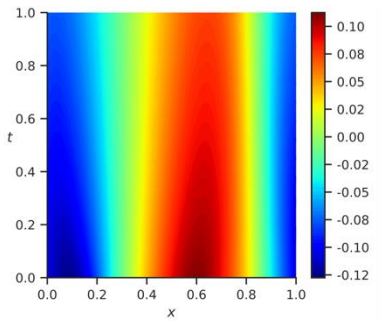


Burgers' Equation

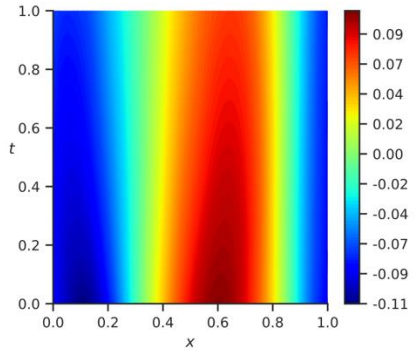
Source
(Input function)



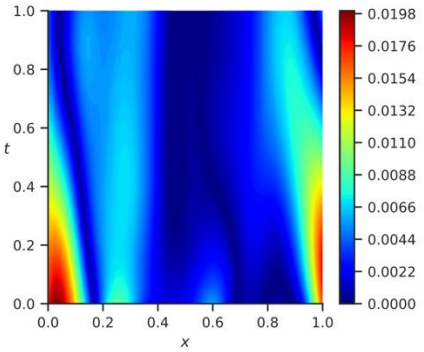
Solution
(Ground Truth)

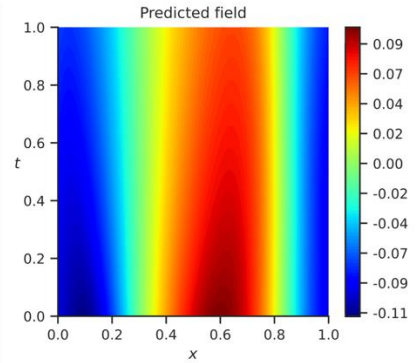


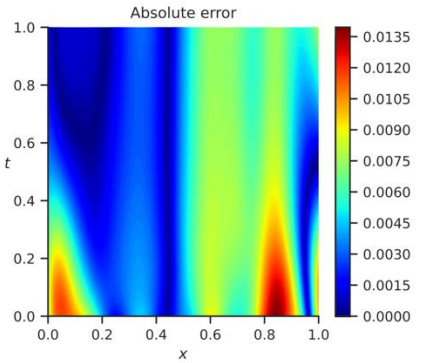
Prediction

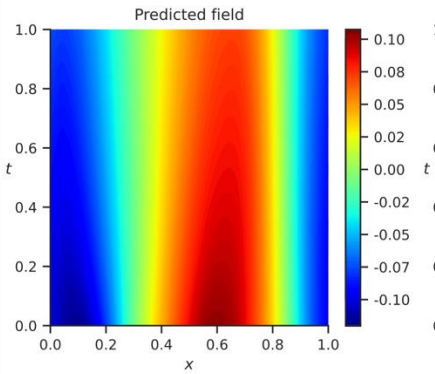


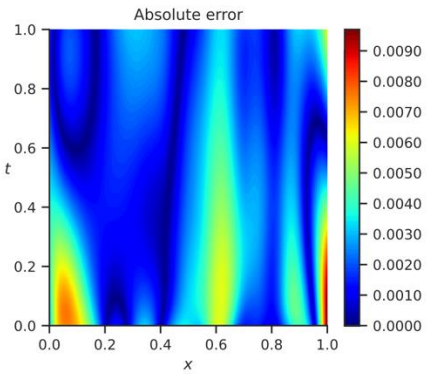
Error



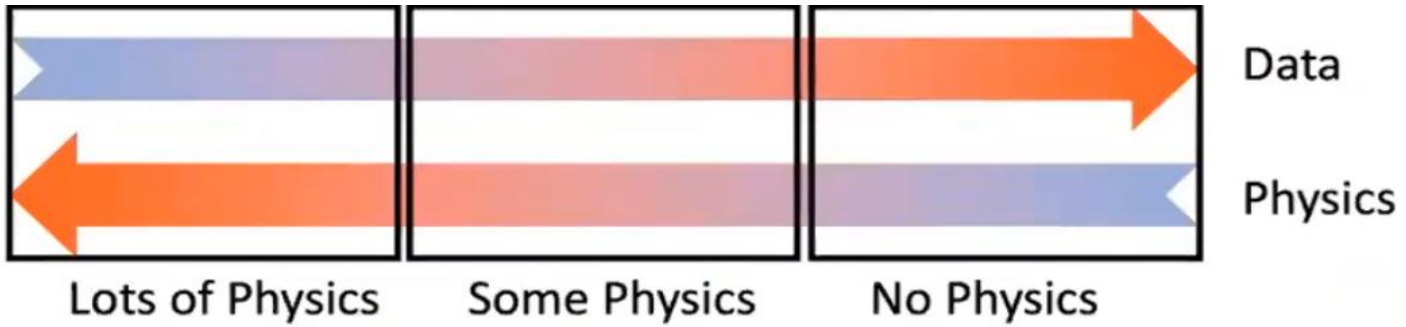








Key Takeaways



- These methods have a niche in real world problems, where partially physics in known and some measurements of quantities of interest are available.
- These methods are best implemented when complemented with mature methods like FEM.
  - Heterogenous multiscale modeling
  - Hybrid fast solvers
- These frameworks offer a possibility to seamlessly blend data and physics.

Acknowledgement








Group Website

Thank you!