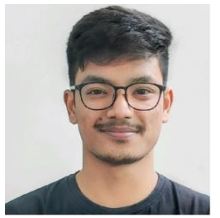
Scientific Machine Learning for Real-Time Physical System Inference: Bridging Physics and Observation

Somdatta Goswami
Civil and Systems Engineering
Johns Hopkins University

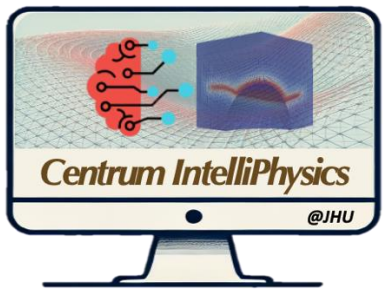
JOHNS HOPKINS UNIVERSITY
U.S. National Science Foundation
U.S. DEPARTMENT OF ENERGY

Centrum IntelliPhysics

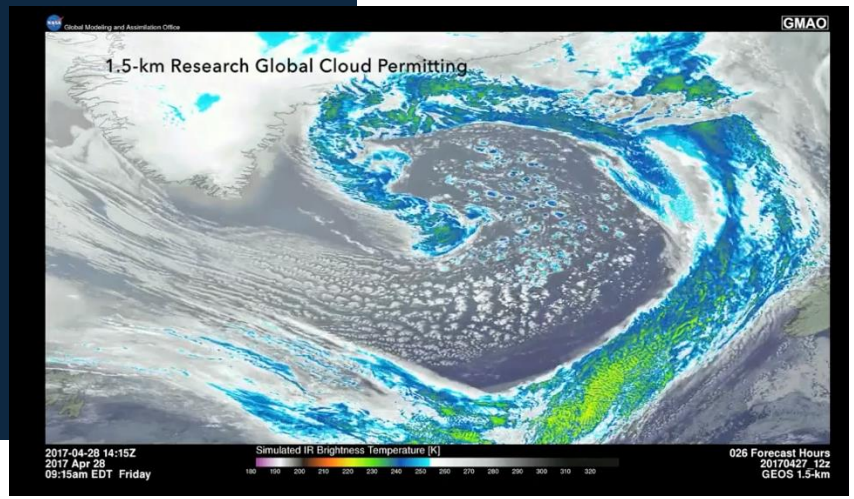
- Mission: Develop machine learning tools to accelerate engineering innovation
- Focus: Physics-Informed Machine Learning
  - Efficient training strategies for neural operators
  - Developing hybrid solvers (operators + solvers)
- Applications: Multiscale Modeling in Materials, Engineering and Biomedical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

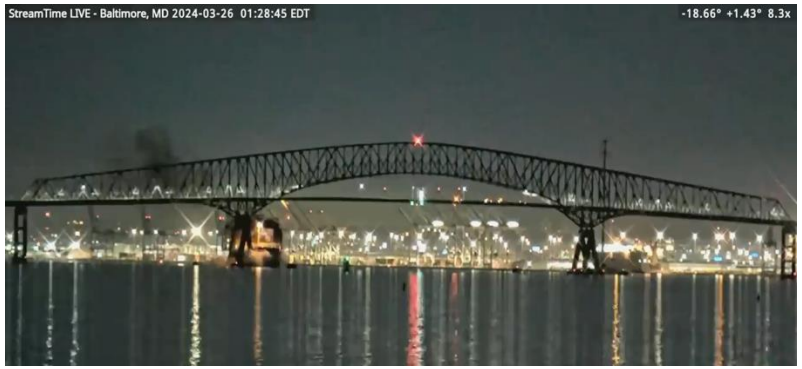


Processes of Nature



Evolution of an Icelandic Low in the North Atlantic Ocean over three days
Source: https://www.nas.nasa.gov/SC17/ (NASA)



Collapse of the Francis Scott Key Bridge in Baltimore on March 26, 2024
Source: https://www.youtube.com/watch?v=2kw1hH5XCYU

Bhuj Earthquake in India in 2001
https://www.youtube.com/shorts/SkNaxcxVz98

Physics-based Models

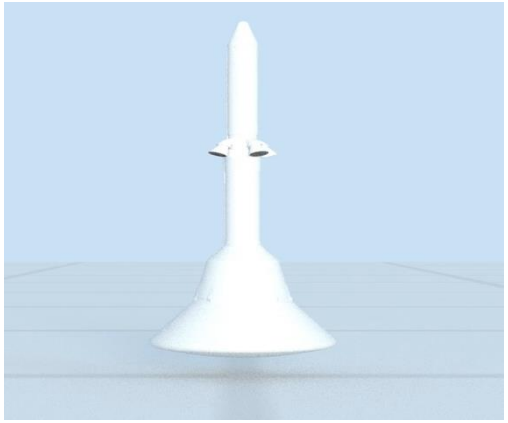
Can represent the Processes of Nature

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

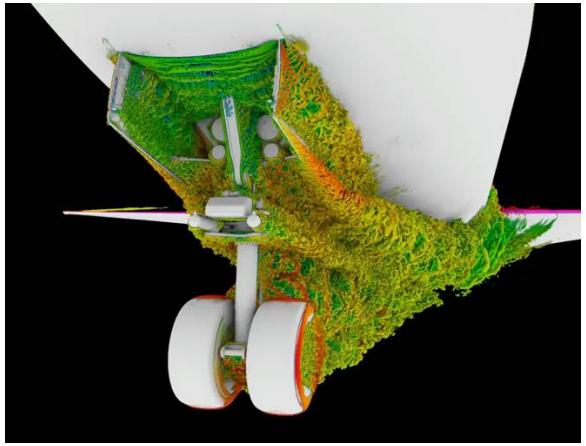
  To model earthquake: $m\frac{d^2u}{dt^2} + \mathrm{k}\frac{du}{dt} + F_0 = 0$

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

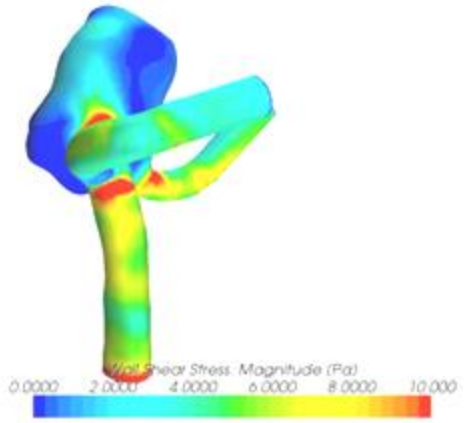
- Computational Mechanics helps us simulate these equations.



Simulation of Orion Spacecraft Launch Abort System (NASA Ames)

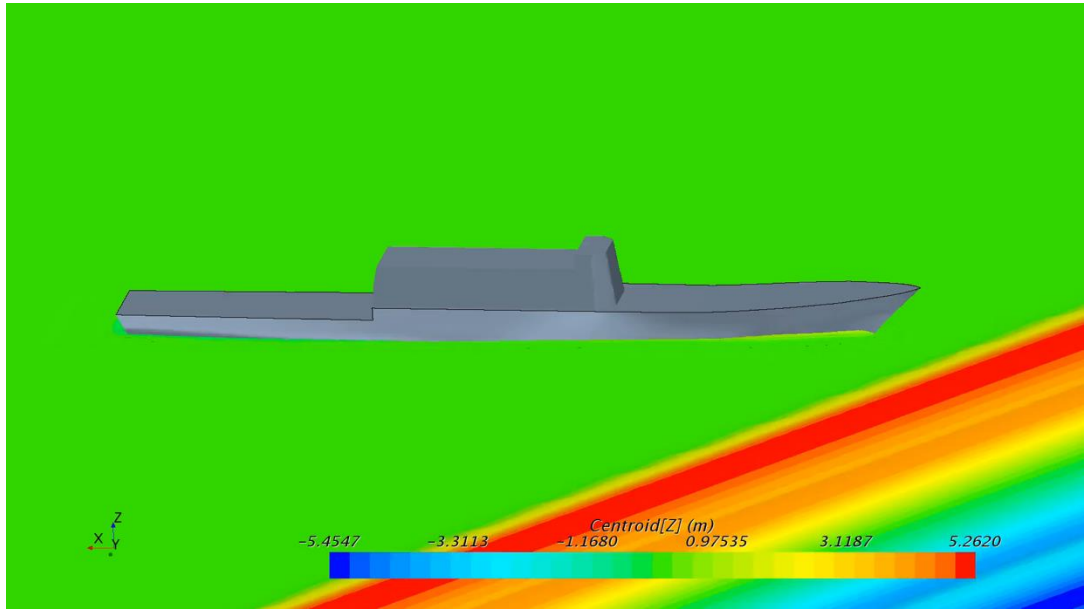


Detailed flow around an Aircraft's landing gear (NASA Ames)

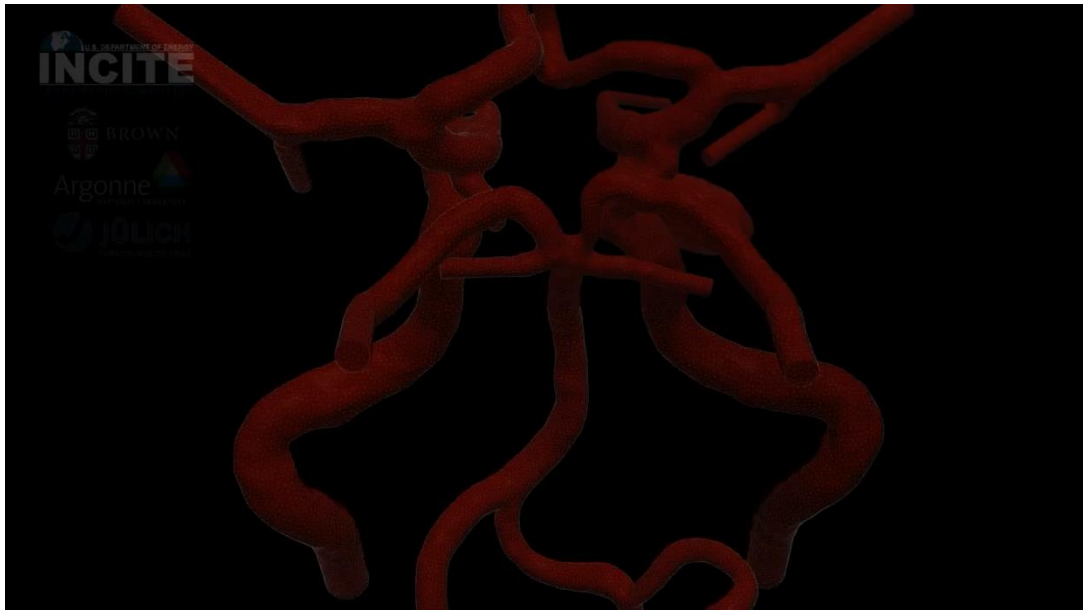


CFD Simulation of a Patient-Specific Intracranial Aneurysm

Simulation Real Scenarios

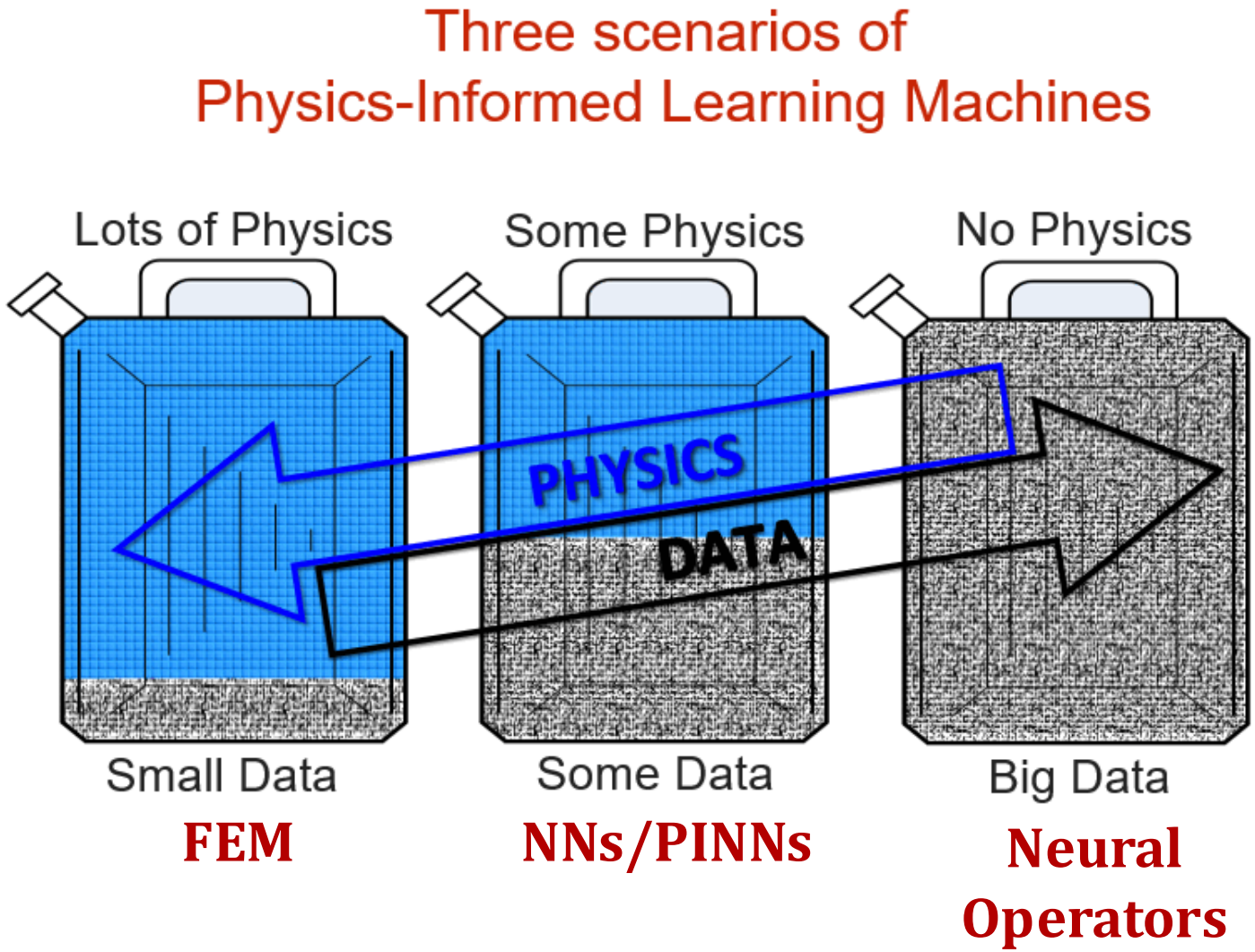


Courtesy: MIT, Michael Triantafyllou



Courtesy: Argonne National Lab

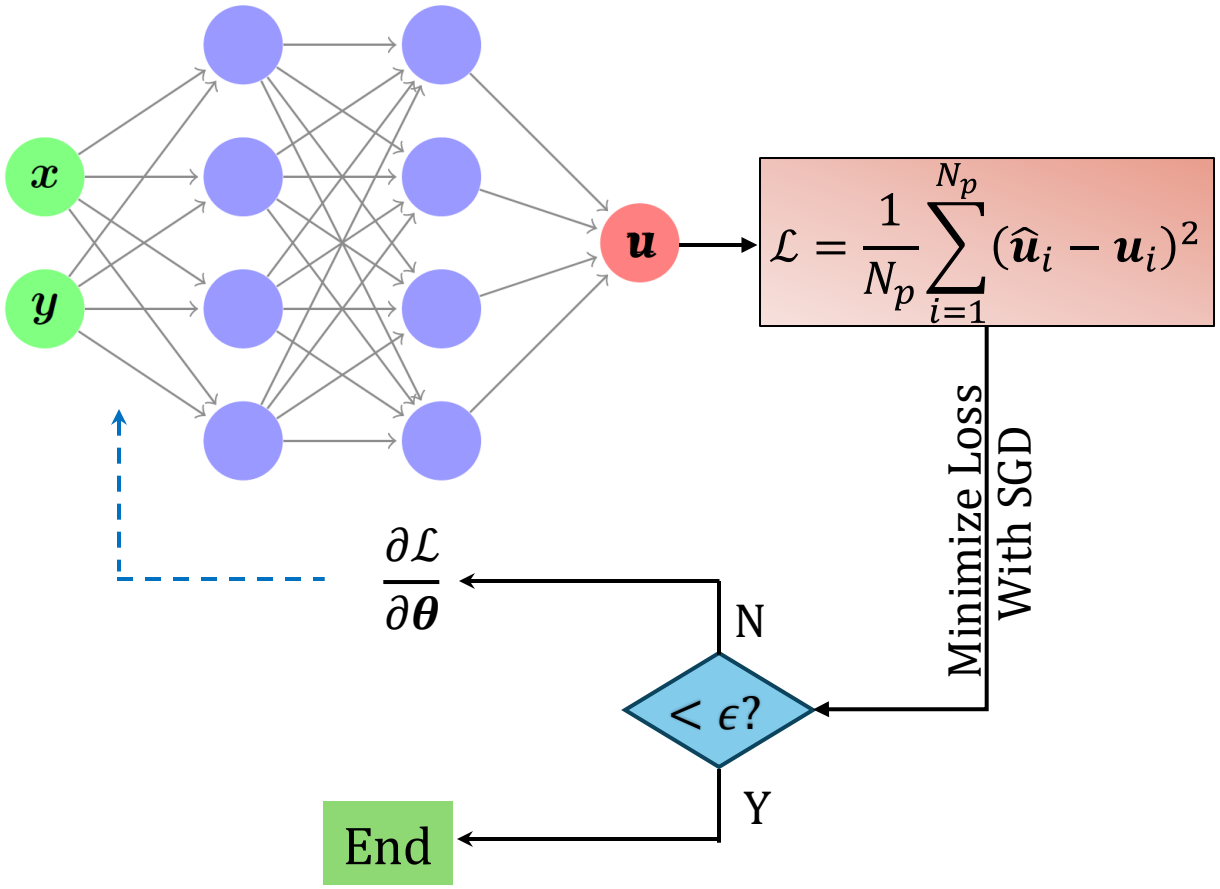
Data + Laws of Physics



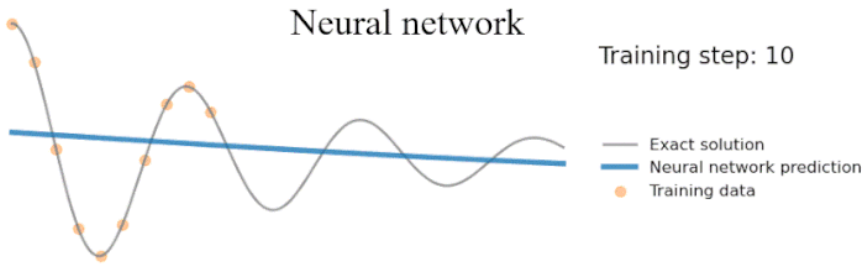
Training data-driven NNs

$$\mathbb{L}(\mathcal{X}, \lambda, \boldsymbol{u}(\boldsymbol{x}); \boldsymbol{\theta}) = g$$
$$\boldsymbol{x} = (x, y) \in \Omega, \lambda \in D, \lambda = 1$$

$\hat{\boldsymbol{u}}$: Predicted solution
$\boldsymbol{\theta}$: Weights and biases
N_p: # of collocation points
$(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}))$: Labelled dataset



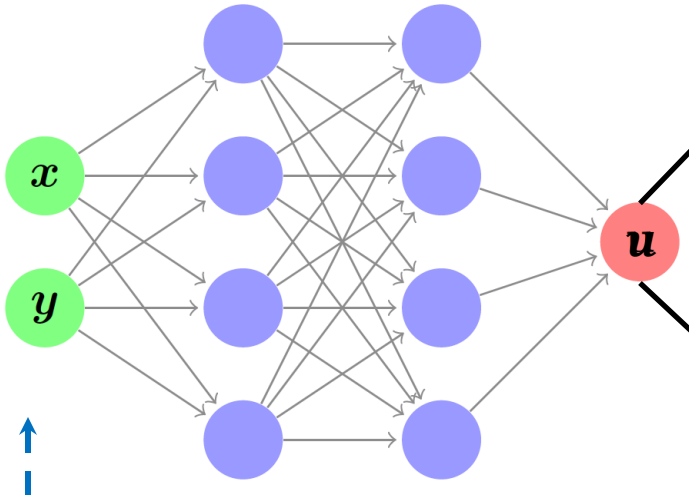
What is the problem with data-driven?



Inspired by Ben Moley blog 2021

Physics-Informed Neural Networks (PINNs)

$$\mathbb{L}(\mathcal{X}, \lambda, \boldsymbol{u}(\boldsymbol{x}); \boldsymbol{\theta}) = g$$
$$\boldsymbol{x} = (x, y) \in \Omega, \quad \lambda \in D, \lambda = 1$$



$\boldsymbol{u}$

$$\mathcal{L}_{ibc} = \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} (\hat{\boldsymbol{u}}_i - \boldsymbol{u}_i)^2 + \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} (\hat{\boldsymbol{u}}_i - \boldsymbol{u}_i)^2$$

$\frac{\partial \boldsymbol{u}}{\partial x}$ ⋮ $\frac{\partial^2 \boldsymbol{u}}{\partial y^2}$

$$\mathcal{L}_{pde} = \frac{1}{N_p} \sum_{i=1}^{N_p} (\mathbb{L}_{\boldsymbol{i}} - \mathrm{g})^2$$

Derivatives computed using AD

$$\mathcal{L} = \mathcal{L}_{ibc} + \mathcal{L}_{pde}$$

Minimize Loss using stochastic gradient optimizer

$< \epsilon$?

N

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

Y

End

$\hat{\boldsymbol{u}}$: Predicted solution

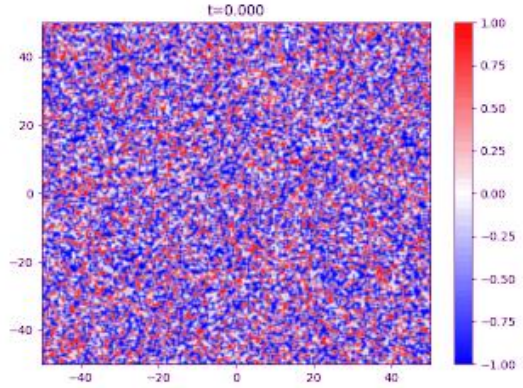
θ : Weights and biases

$N_{ic/bc}$: Initial conditions and boundary conditions, $N_{ic} + N_{bc} \ll N_p$

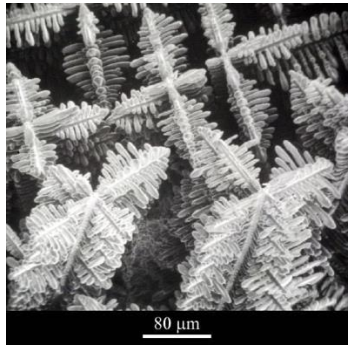
Phase-field modeling for physical systems

- widely used in material science for morphological evolution. The interfaces for different phases can be easily tracked without any specific efforts.

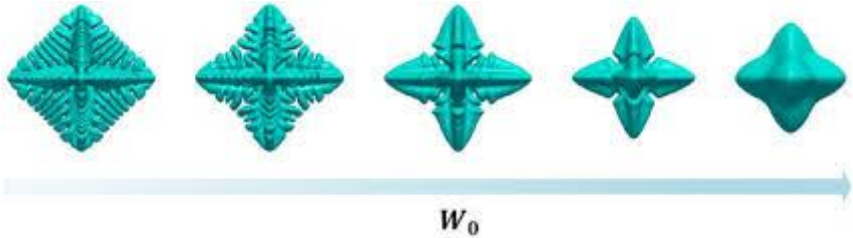
Concentration evolution



Dendritic growth

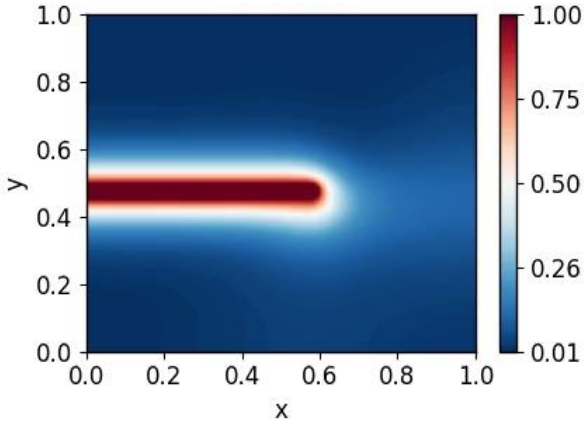


See R. Glardon, W. Kurz,(1981)



Chen W, Zhao Y, Yang S, et al. (2021)

Fracture Modeling



Challenges in Phase-field modeling

- Computation is very long and strenuous simulations even for a 2D example.
- Solution accuracy is dependent on discretization.

Can PINNs be used to accelerate it?

- Can be, still difficult.
- Neural network can approximate continuous functions.
- Might not be as accurate as FEM.

Our Proposed Frameworks

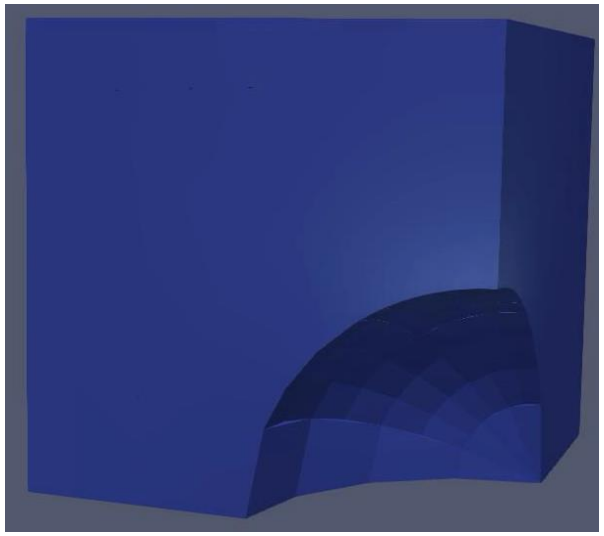
Somdatta Goswami, Cosmin Anitescu, Souvik Chakraborty, and Timon Rabczuk. "Transfer learning enhanced physics informed neural network for phase-field modeling of fracture." *Theoretical and Applied Fracture Mechanics* 106 (2020): 102447.

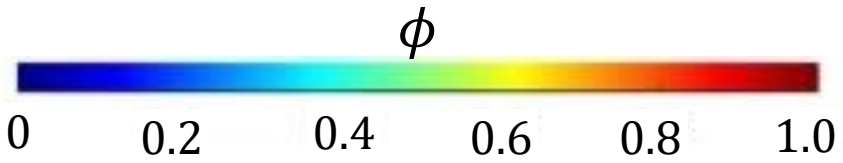


Wei Wang, Tang Paai Wong, Haihui Ruan, and Somdatta Goswami. "Causality-Respecting Adaptive Refinement for PINNs: Enabling Precise Interface Evolution in Phase Field Modeling." *arXiv preprint arXiv:2410.20212* (2024).



Phase-field fracture modeling



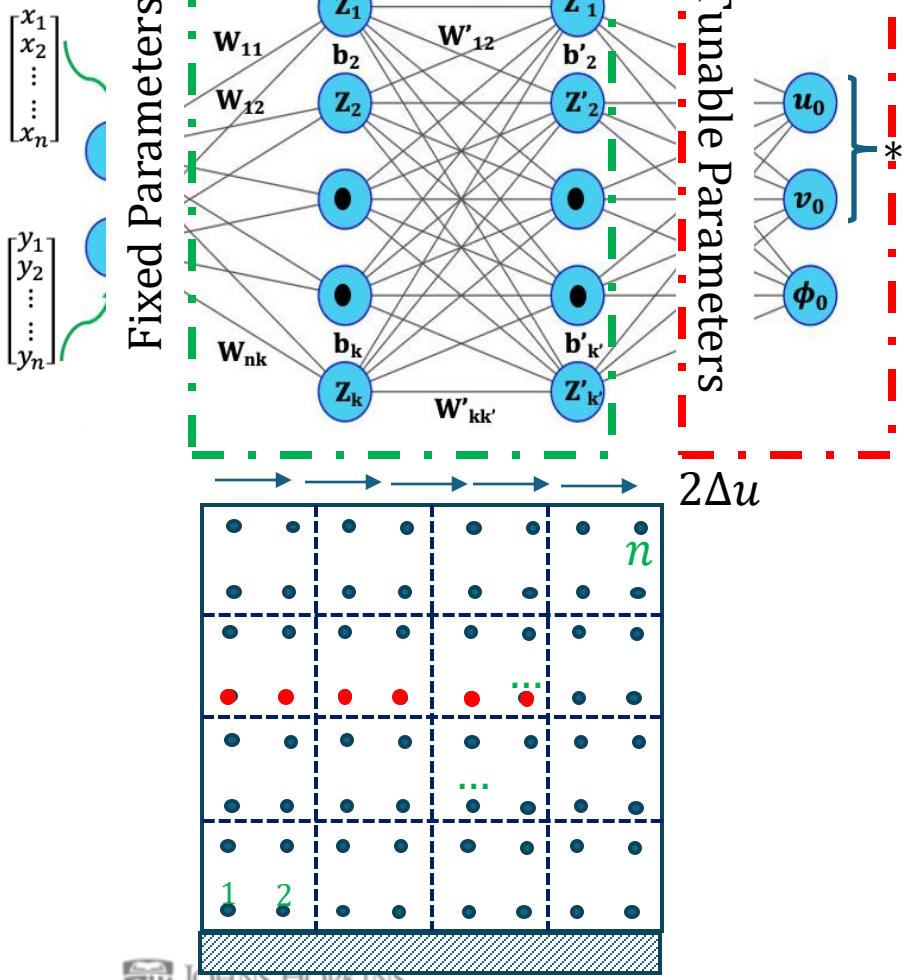


- The thickness of the damaged region controlled using l_s.
- Regularized total potential energy: [Bourdin et al. '00, CMAME]

$$\mathcal{E}(u,\phi) = \underbrace{\int_\Omega g(\phi)\psi_e(u)\ d\Omega}_{\text{Elastic energy}} + \underbrace{\int_\Gamma \mathcal{G}_c \left(\frac{1}{2l_s}\phi^2 + \frac{l_s}{2}|\nabla\phi|^2\right) d\Gamma}_{\text{Volumetric approximation of fracture energy}} + \underbrace{\int_\Omega g''(\phi)\boldsymbol{H_0}(x,t)d\Omega}_{\text{To impose Irreversibility}}$$

- Displacement-controlled fracture

Variational Energy based PINNs (VE-PINNs)



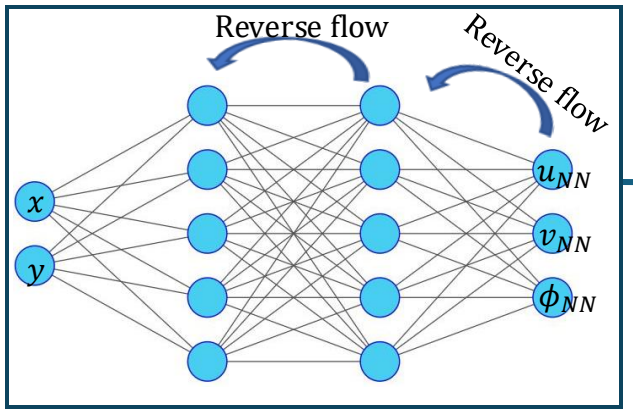
$$* C(x,y) + \tilde{u} = \begin{Bmatrix} u_{NN} \\ v_{NN} \\ \{\phi_{NN}\} \end{Bmatrix}$$

$f(x, y, W, b, \sigma)$

Back Propagation: $\theta_{i+1} = \theta_i - \eta \, {\partial J(\theta)}/{\partial \theta}$

Variational Energy, $\mathcal{E}$

$$\int g(\phi)\psi_0 \, d\Omega + \int \frac{G_c}{2l_0}(\phi^2 + l_0^2 |\nabla\phi|^2) \, d\Omega + \int g''(\phi) \boldsymbol{H_0}(x,t) d\Omega = J_{min}(\theta)$$



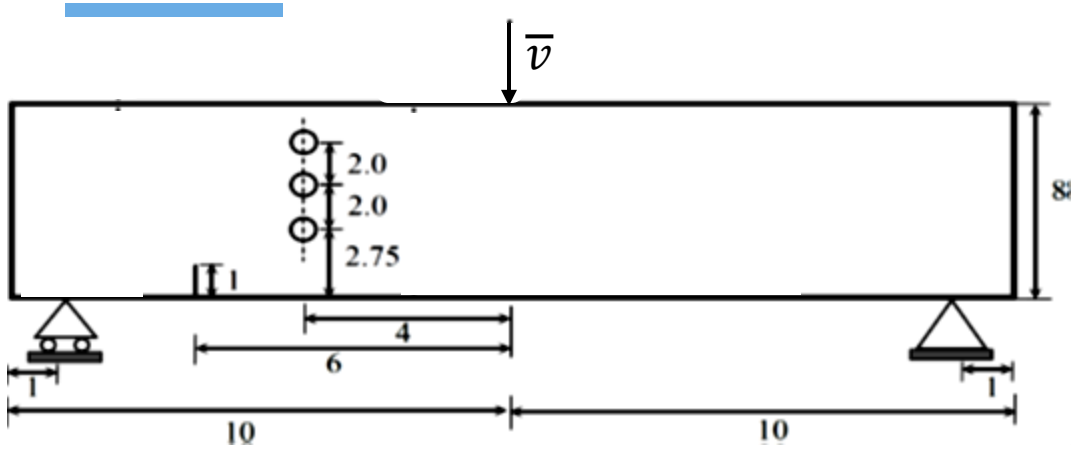
Automatic differentiation

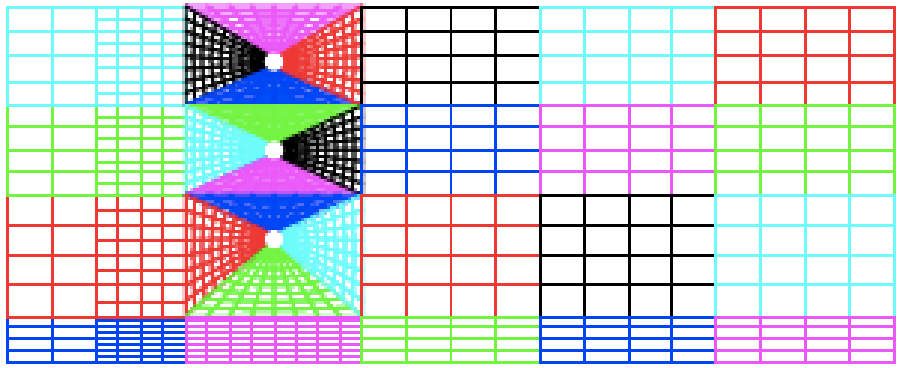
$$\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}, \frac{\partial \phi}{\partial x}$$

$f(x, y, W, b, \sigma')$

Goswami S, Anitescu C, Chakraborty S and Rabczuk T. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *TAFMEC*

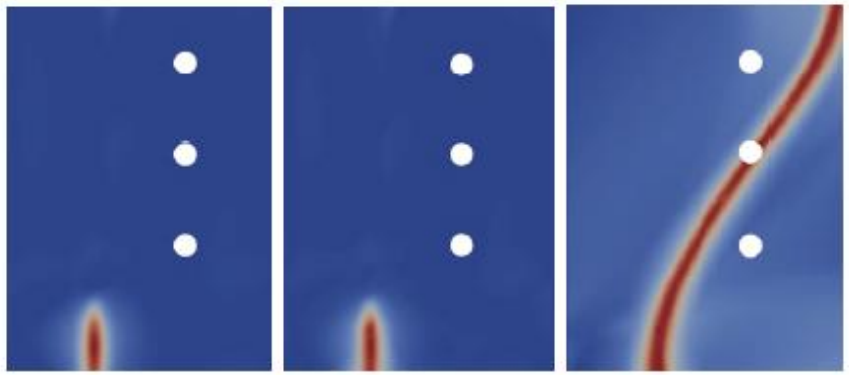
Perforated and asymmetrically notched beam





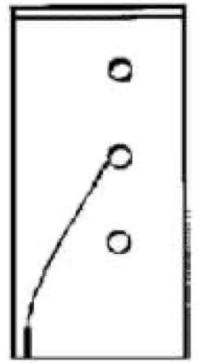
- $\Delta v = 1e-2$ mm
- $l_s \quad = 0.025$ mm
- 1184 elements, 25 Gauss points/element.
- $u = \frac{w_2}{(w_2+1)}\hat{u}$
- $v = \frac{w_1 w_2 w_3}{(w_1+1)(w_2+1)(w_3+1)}\hat{v} + \frac{y}{8}\nabla v$
- $w_1 = (x-1)^2 + y^2, w_2 = (x-19)^2 + y^2$
- $w_3 = (x-10)^2 + (y-8)^2$

Results from VE-PINNs



$u = 0$ $u = 10^{-2}$ u = 0.16

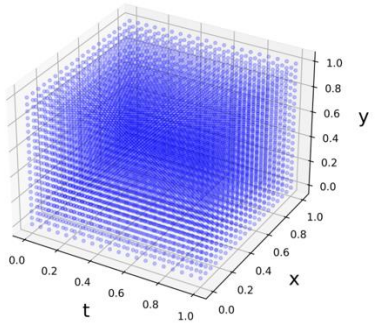
Experimental



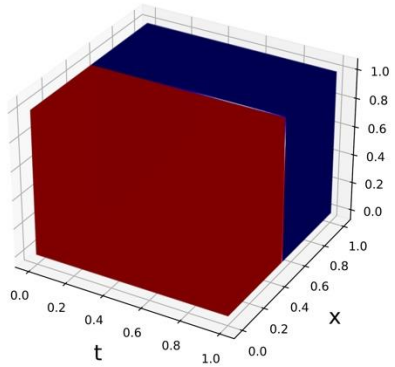
Allen-Cahn Equation in PINNs

$$\frac{\partial \phi}{\partial t} = -L_\sigma \left(g'(\phi) - \kappa \nabla^2 \phi \right)$$ (Static interface modeling)

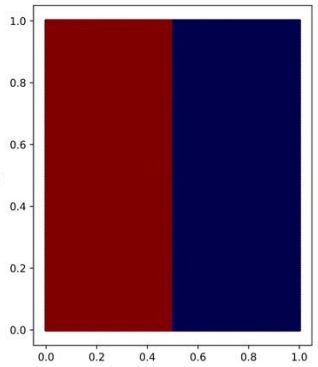
20 ×20×20 points



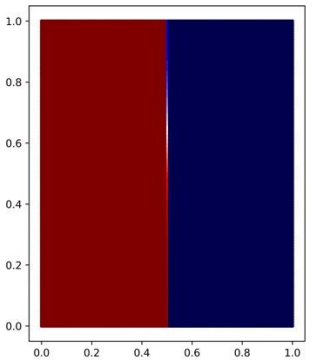
3D view



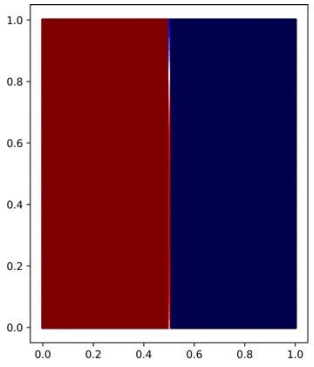
0s



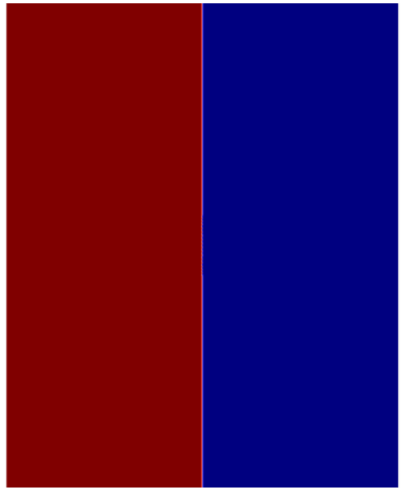
0.5s



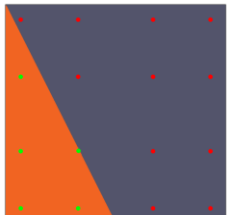
1s



Numerical solver:

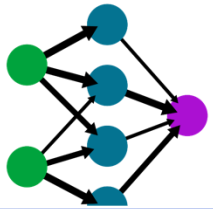


Residual-based Adaptive (RBAR)



Generate points

- Uniform collocation points



Solve

- Using PINNs
- For single Δt

Estimate

- Elastic residual (ER)

Mark

- Top ρ% elements contributing to max error in ER

Zone?

N

Next step

Y

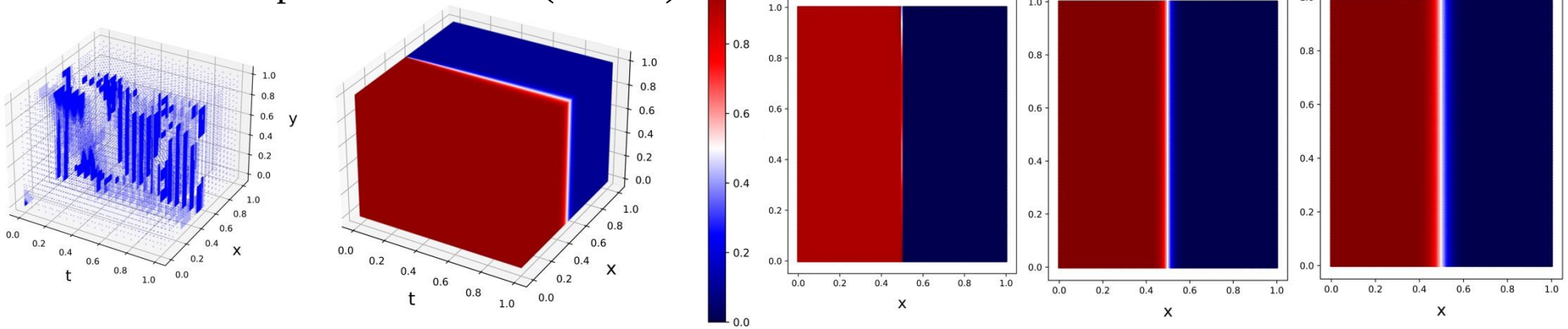
Refine the zone

Goswami S, Anitescu C and Rabczuk T, 2020. Adaptive fourth-order phase field analysis using deep energy minimization. *TAFMEC*

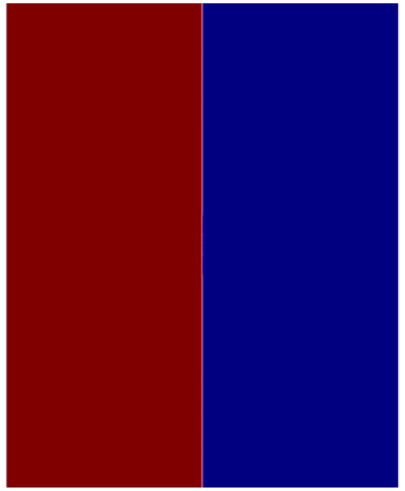
Allen-Cahn Equation in PINNs

$$\frac{\partial \phi}{\partial t} = -L_\sigma \left(g'(\phi) - \kappa \nabla^2 \phi \right)$$ (Static interface modeling)

Residual based adaptive refinement (**RBAR**)



Numerical solver:



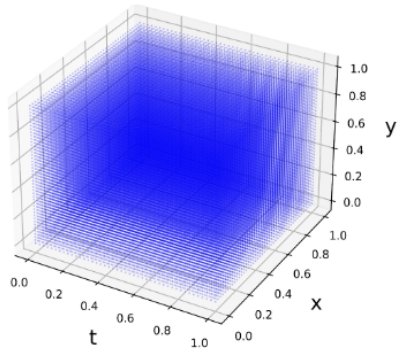
Allen-Cahn (Dynamic Modeling)

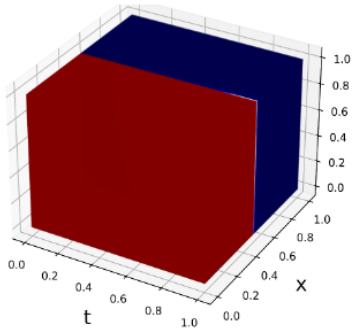
$$\frac{\partial \phi}{\partial t} = -L_\sigma \left(g'(\phi) - \kappa \nabla^2 \phi \right) + L_\eta P'(\phi)$$

Constant driving force at interface

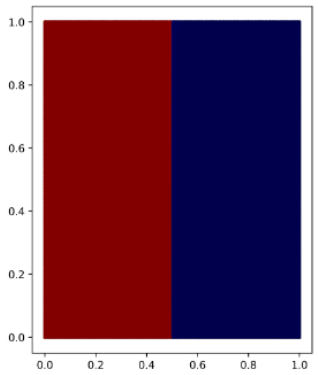
Uniform refinement

44×44×44 collocation points

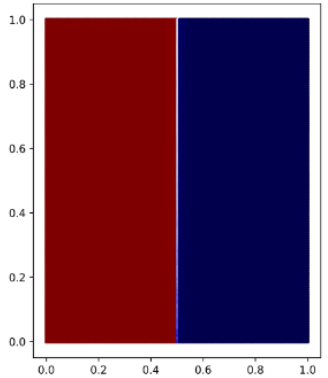




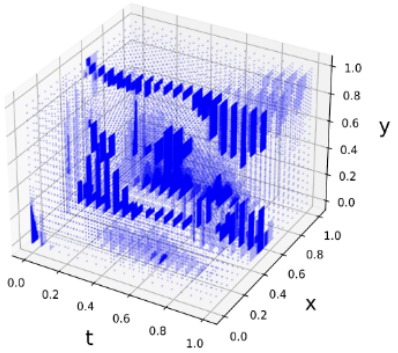
0.5s

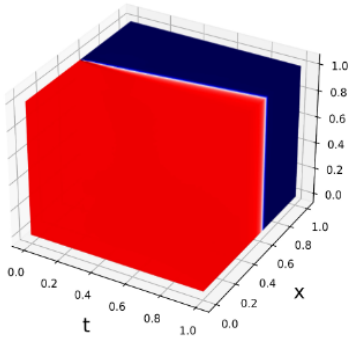


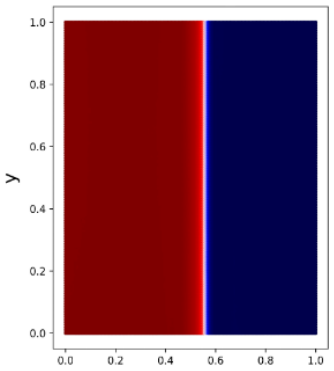
1s

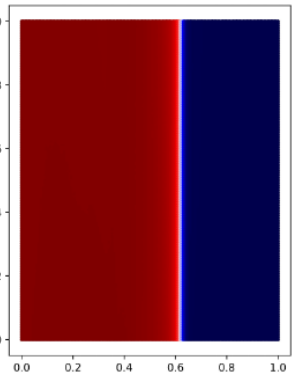


RBAR

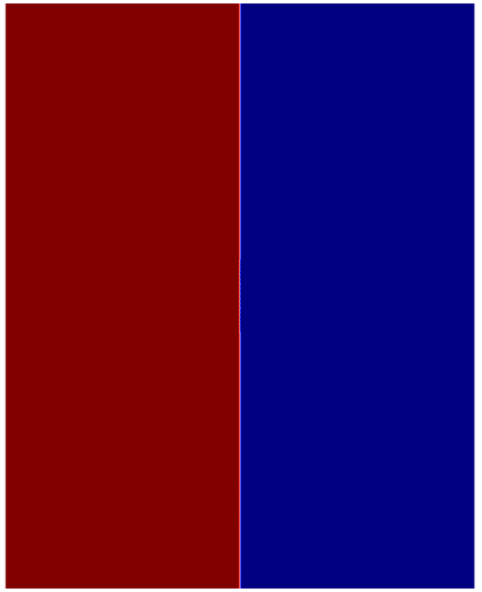






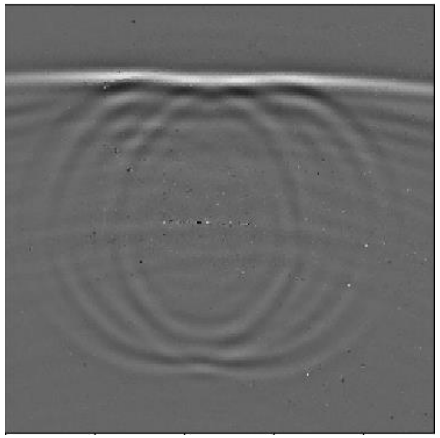


Numerical solver:



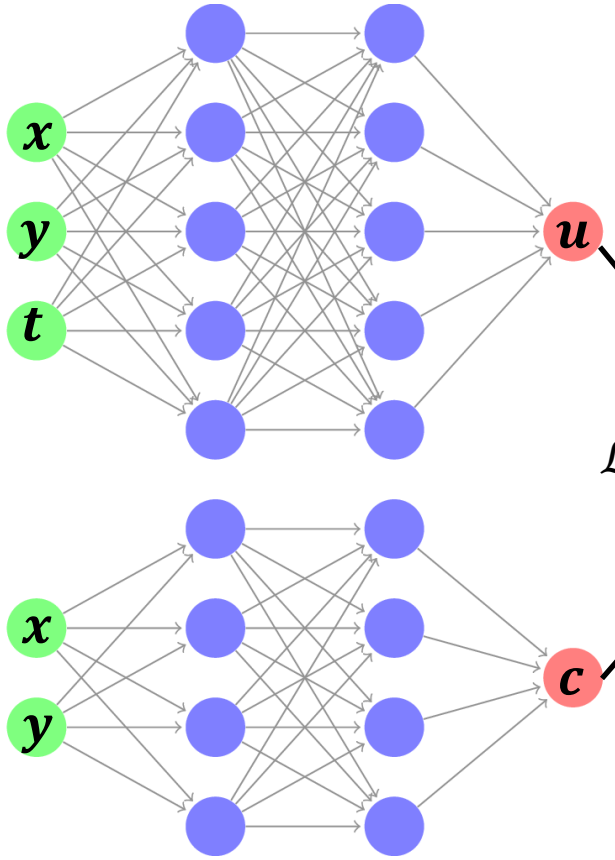
Crack characterization in an aluminum alloy using PINNs

- Objective: Identifying and characterizing a surface breaking crack in a 7075-T651 aluminum alloy metal plate.
- Dataset: Particle displacement induced by surface acoustic waves of 5 MHz at three incidence angle 0°, 45°, and 90°



0° incidence

$$f = \frac{\partial^2 u}{\partial t^2} - c(x,y)^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

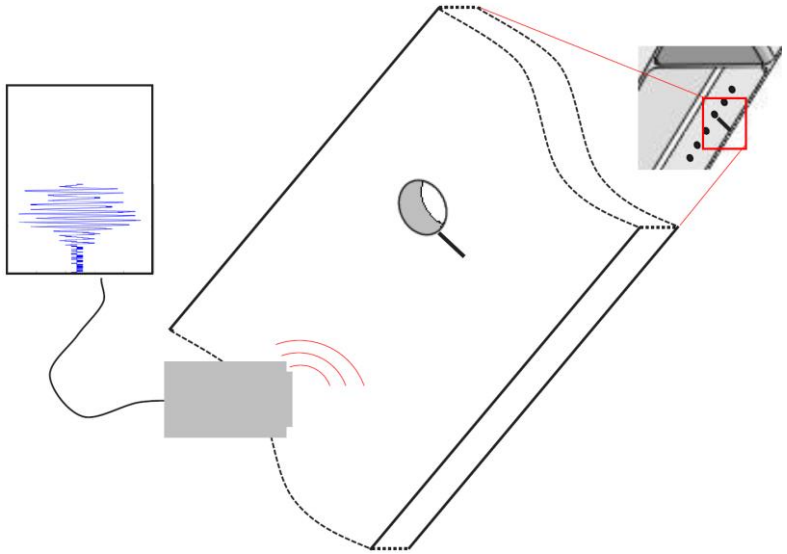


$$\mathcal{L}_{data} = |u_{NN}(x,y,t) - u(x,y,t)|^2$$

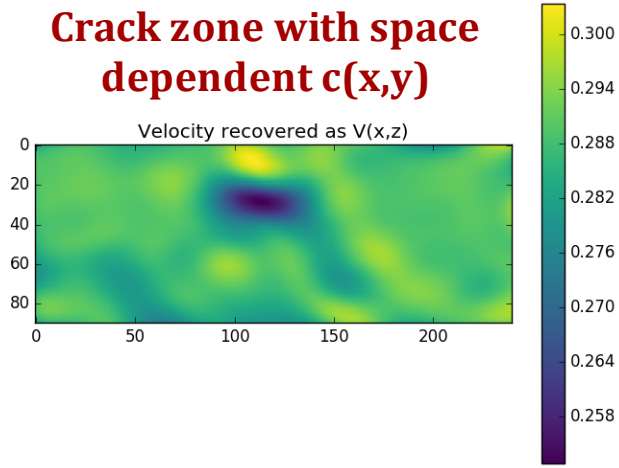
$$\mathcal{L}_{pde} = |\frac{\partial^2 u_{NN}}{\partial t^2} - c_{NN}(x,y)^2\left(\frac{\partial^2 u_{NN}}{\partial x^2} + \frac{\partial^2 u_{NN}}{\partial y^2}\right)$$

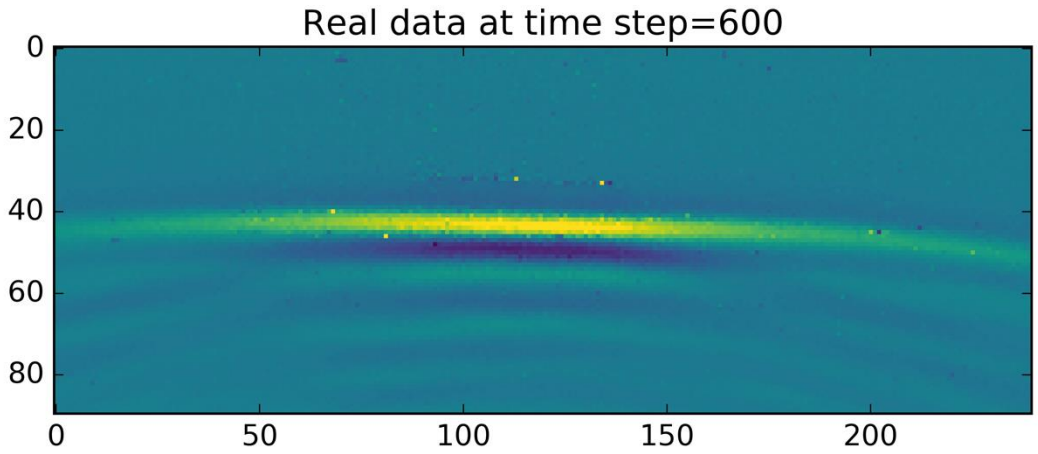
$$\mathcal{L} = \lambda\mathcal{L}_{data} + \mathcal{L}_{pde}$$

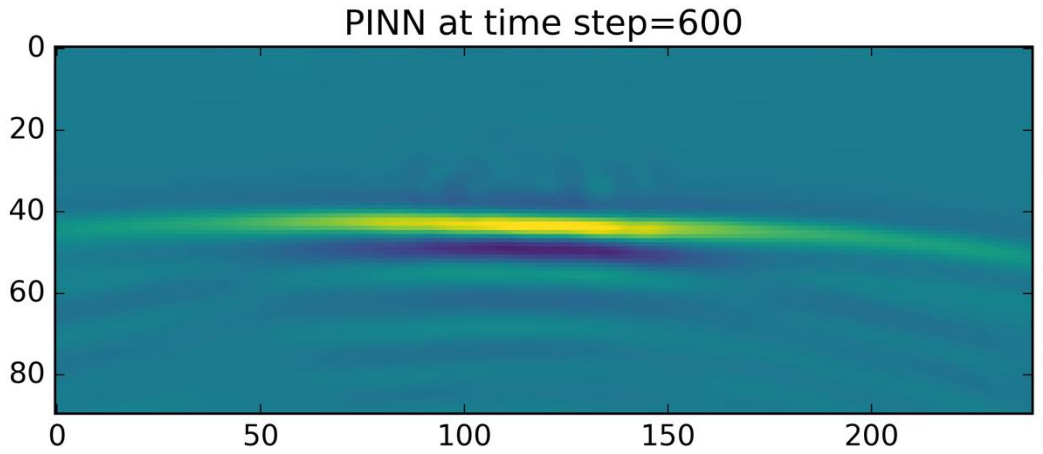
Crack characterization in an aluminum alloy using PINNs



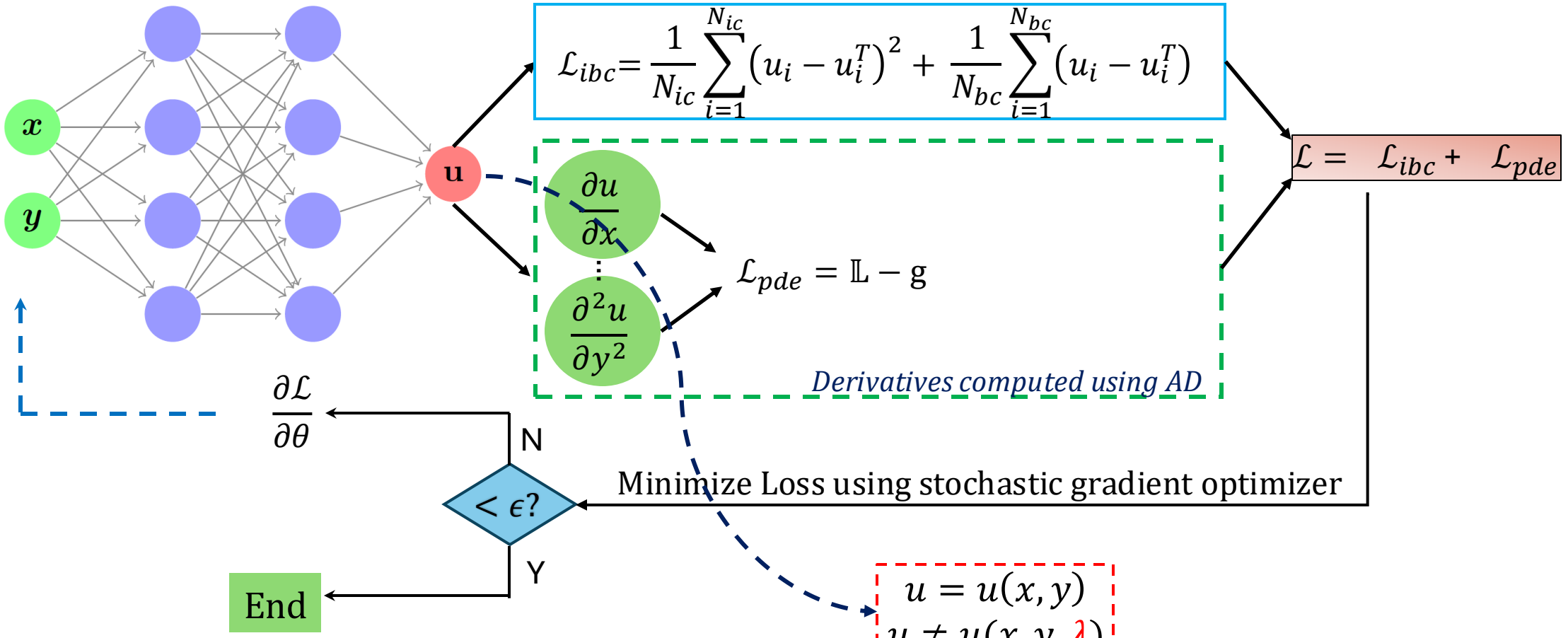
Crack zone with space dependent c(x,y)







Challenges of PINNs



$\mathbb{L}(\mathcal{X}, \boldsymbol{\lambda}, \boldsymbol{u}(\boldsymbol{x}); \boldsymbol{\theta}) = g$

$\boldsymbol{x} = (x, y) \in \Omega, \lambda \in D, \boldsymbol{\lambda} = 1$

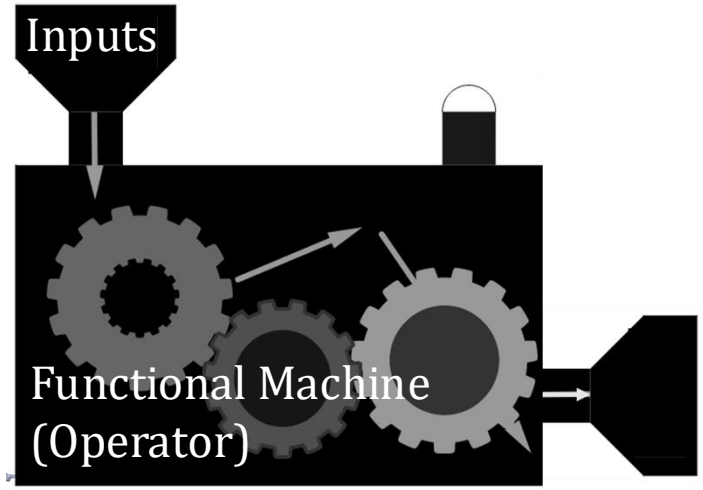
u^T: Ground truth of the solution

θ : Weights and biases

$N_{ic/bc}$: Initial conditions and boundary conditions, $N_{ic} + N_{bc} \ll N_p$

Learning a Neural Operator

$$\mathcal{F} = \{f(\boldsymbol{x}, \lambda_1), f(\boldsymbol{x}, \lambda_2), \dots, f(\boldsymbol{x}, \lambda_n)\}$$



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

$\mathcal{F}, \mathcal{S}$ are infinite dimensional function space

Outputs

$$\mathcal{S} = \{u(\boldsymbol{x}, \lambda_1), u(\boldsymbol{x}, \lambda_2), \dots, u(\boldsymbol{x}, \lambda_n)\}$$

What we know

$$\{\mathcal{F}_n, \mathcal{S}_n\}_{n=1}^{N}$$

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

Task

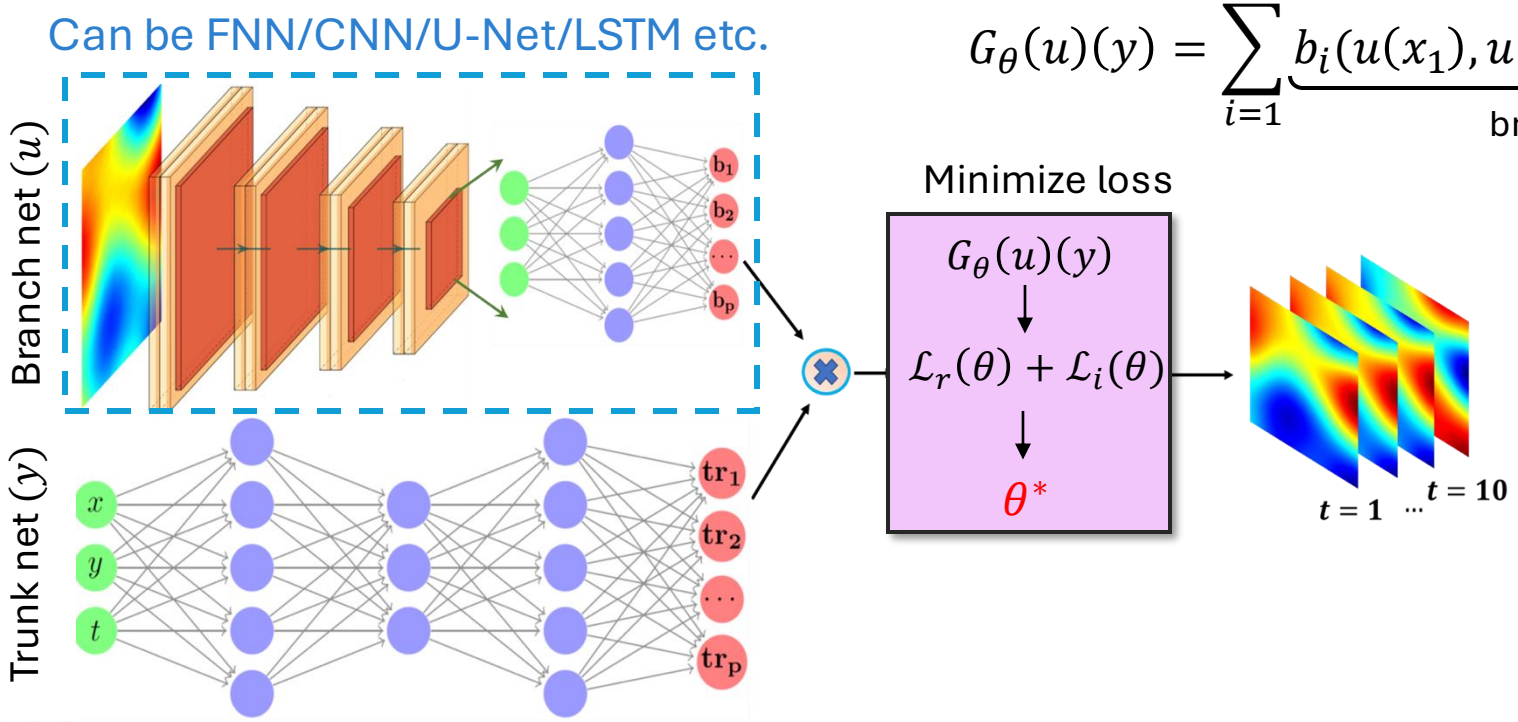
Operator, $\Psi : \mathcal{F} \times \Theta \to \mathcal{S}$ such that $\Psi(., \theta^*) \approx \Phi$

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

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

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



Our Proposed Framework for High-Dimensional Systems

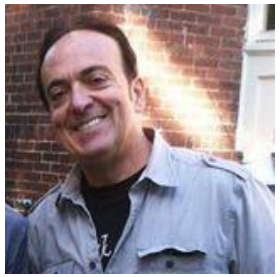
nature communications

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

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

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

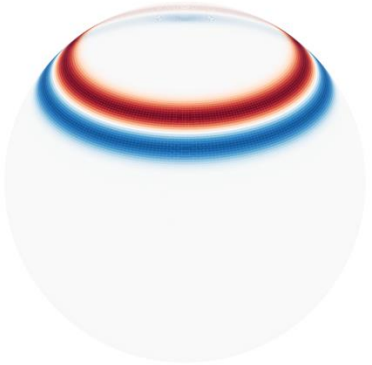
$$h'(\lambda, \phi) = \hat{h}\cos(\phi)\mathrm{e}^{-(\lambda/\alpha)^2}\mathrm{e}^{-[(\phi_2-\phi)/\beta]^2}$$

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

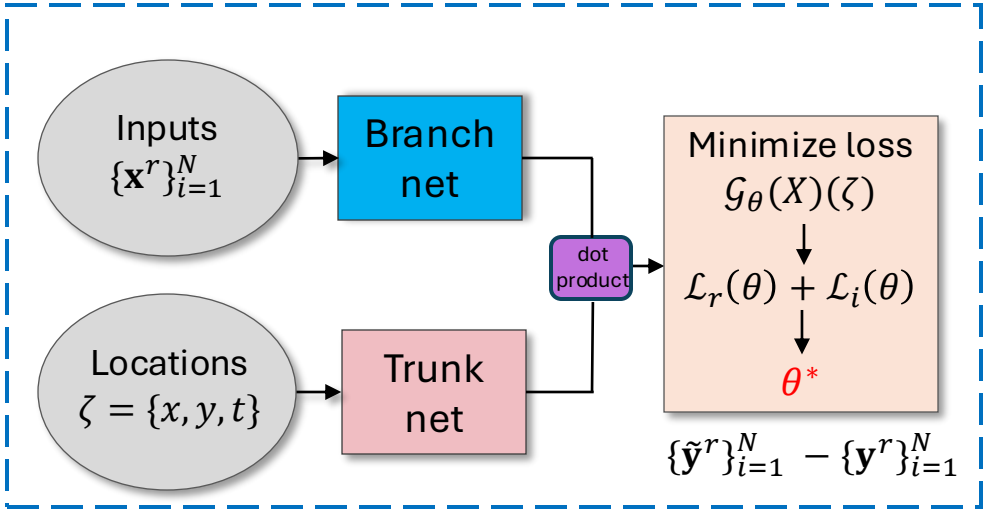
Atmospheric Flow

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

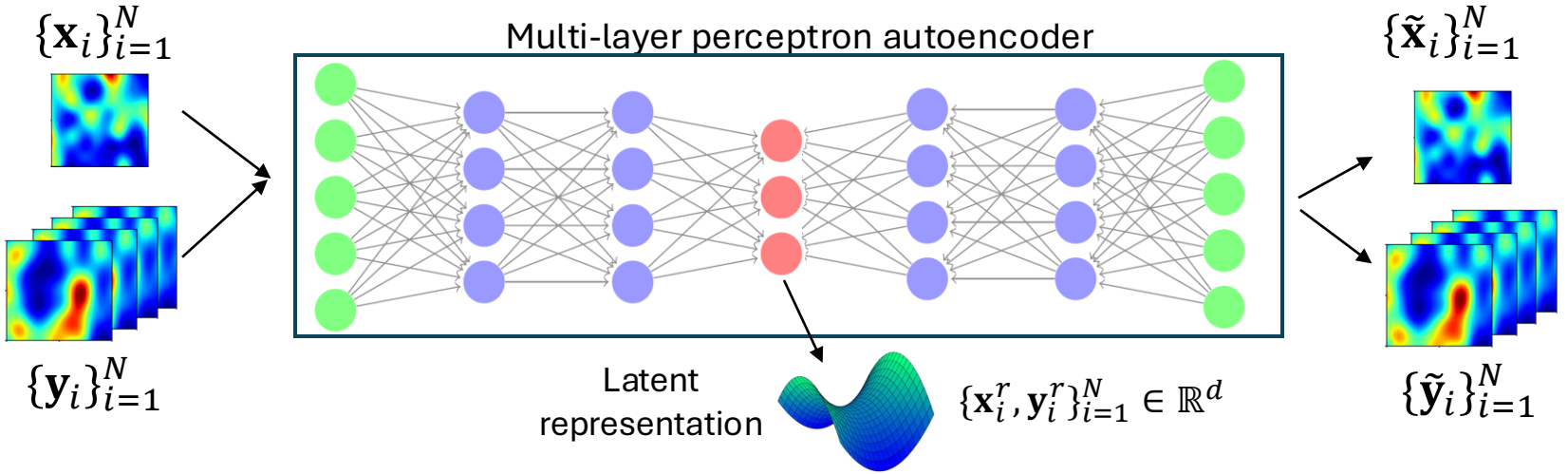


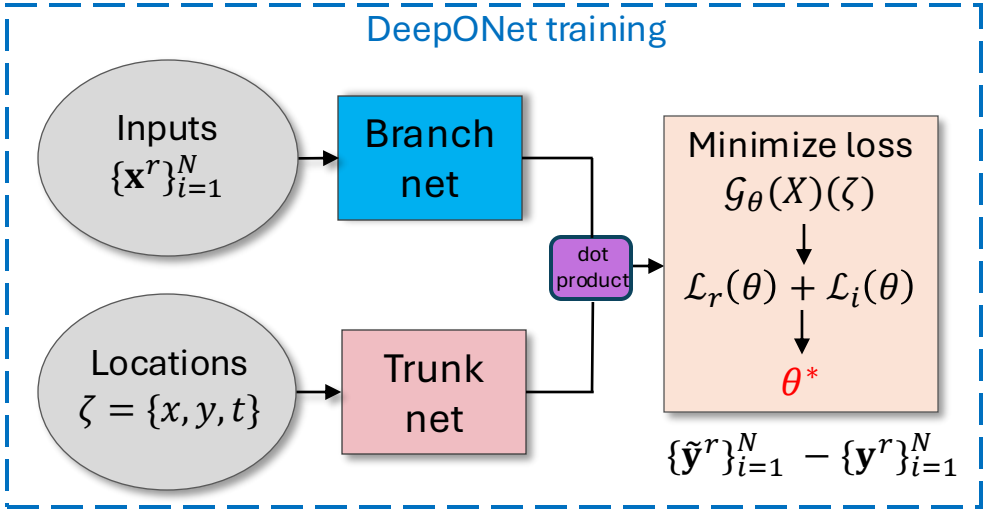
Latent DeepONet for time-dependent PDEs

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

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

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



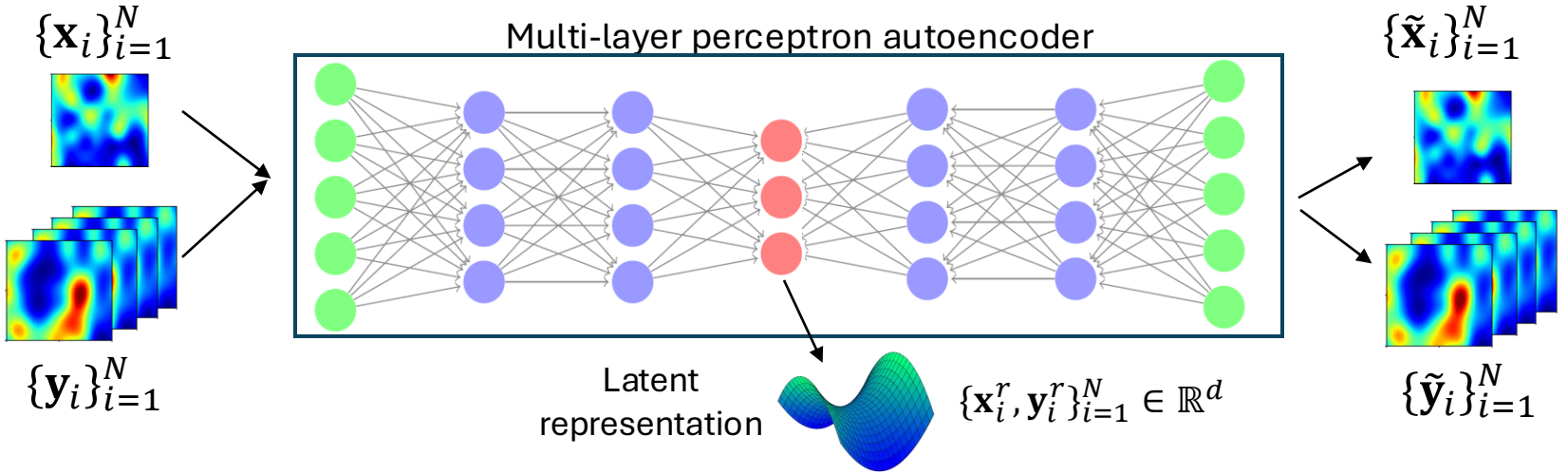


Latent DeepONet for time-dependent PDEs

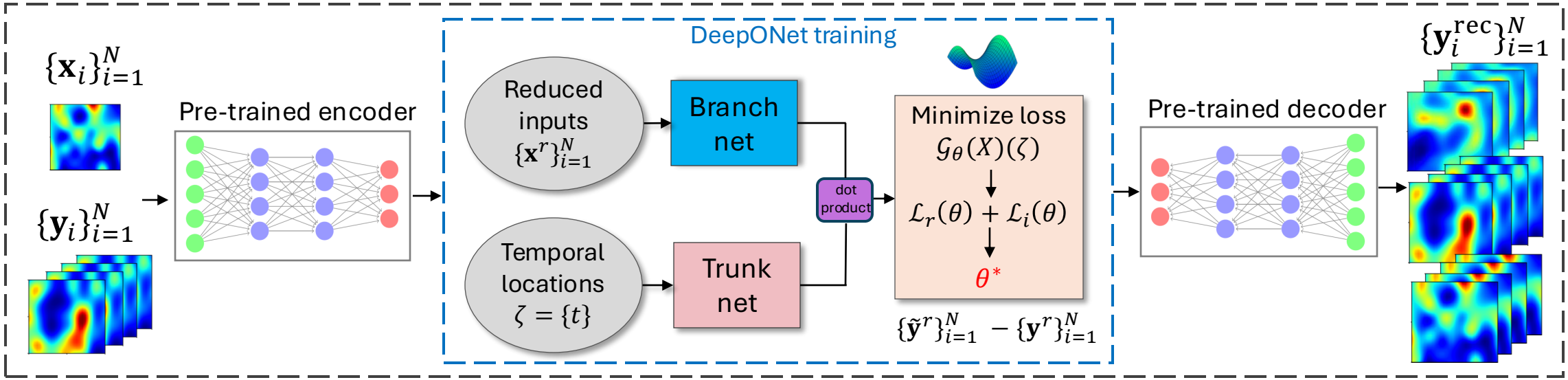
Operator $\mathcal{G}: \mathcal{X} \to \mathcal{Y}$

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

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



Latent DeepONet



MLAE: multi-layer autoencoders
PCA: principal component analysis

Consolidated results

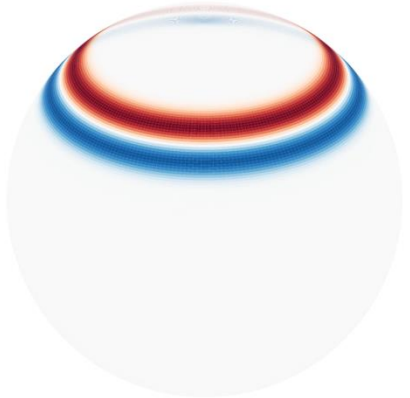
Accuracy of *L-DeepONet for MLAE and PCA*

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

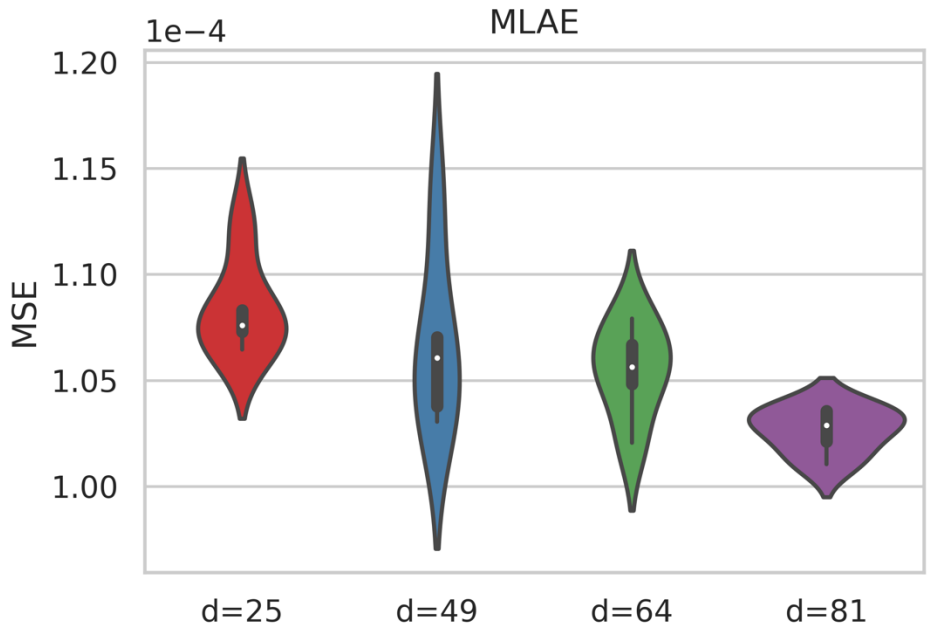
Results

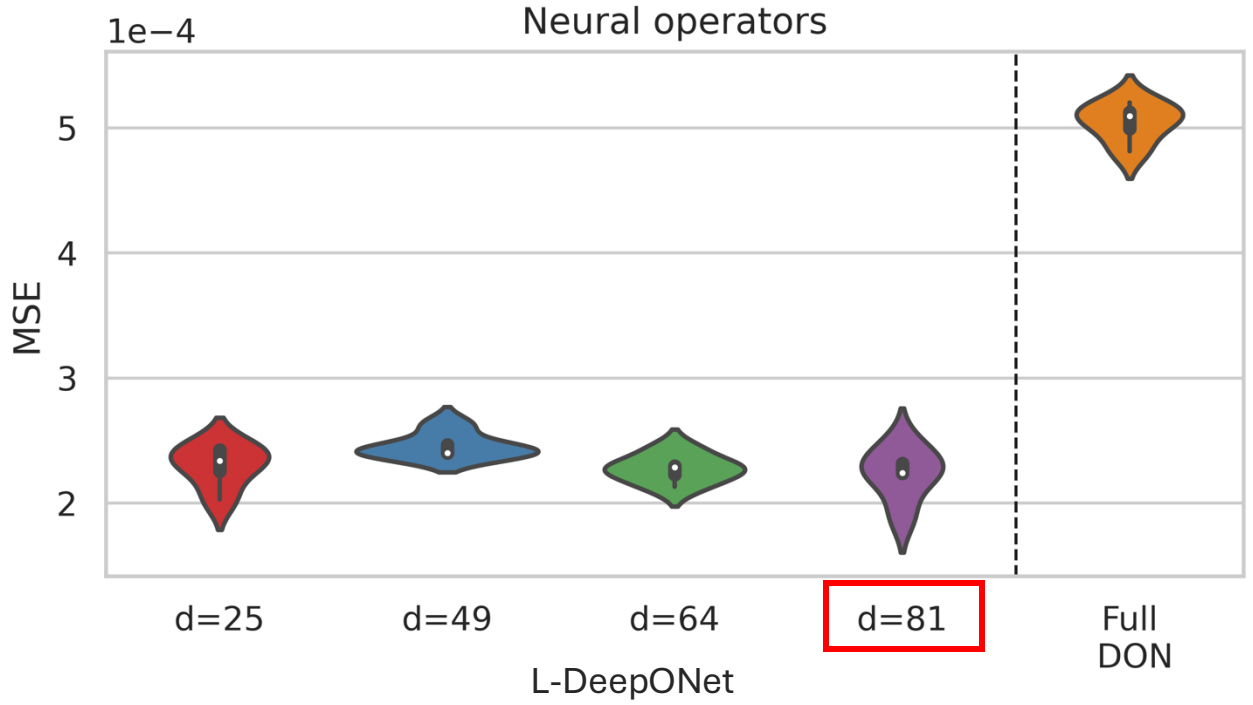
- $\Omega = [0,2\pi]x[0,2\pi]$, $(n_x \text{ x } n_y) = (256x256)$ mesh points
- Output dimensionality: $72x256x256 = 4{,}718{,}592$
- Simulation: $t = [0,360h]$, $\delta t = 0.1\bar{6}h$, Time steps: $n_t = 72$

Training Time (seconds)

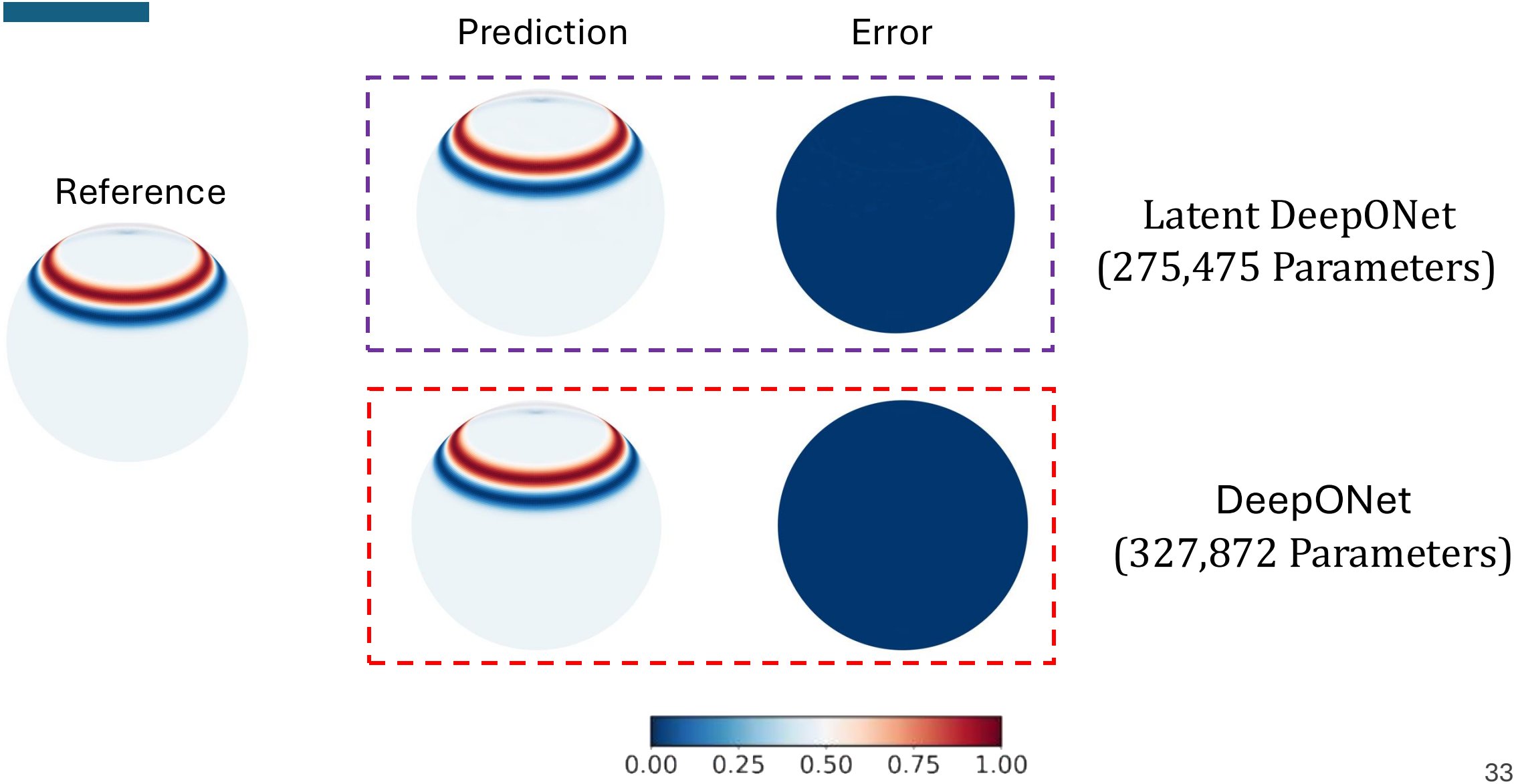
MLAE + Latent DON: 15, 218

Full DON: 379,022

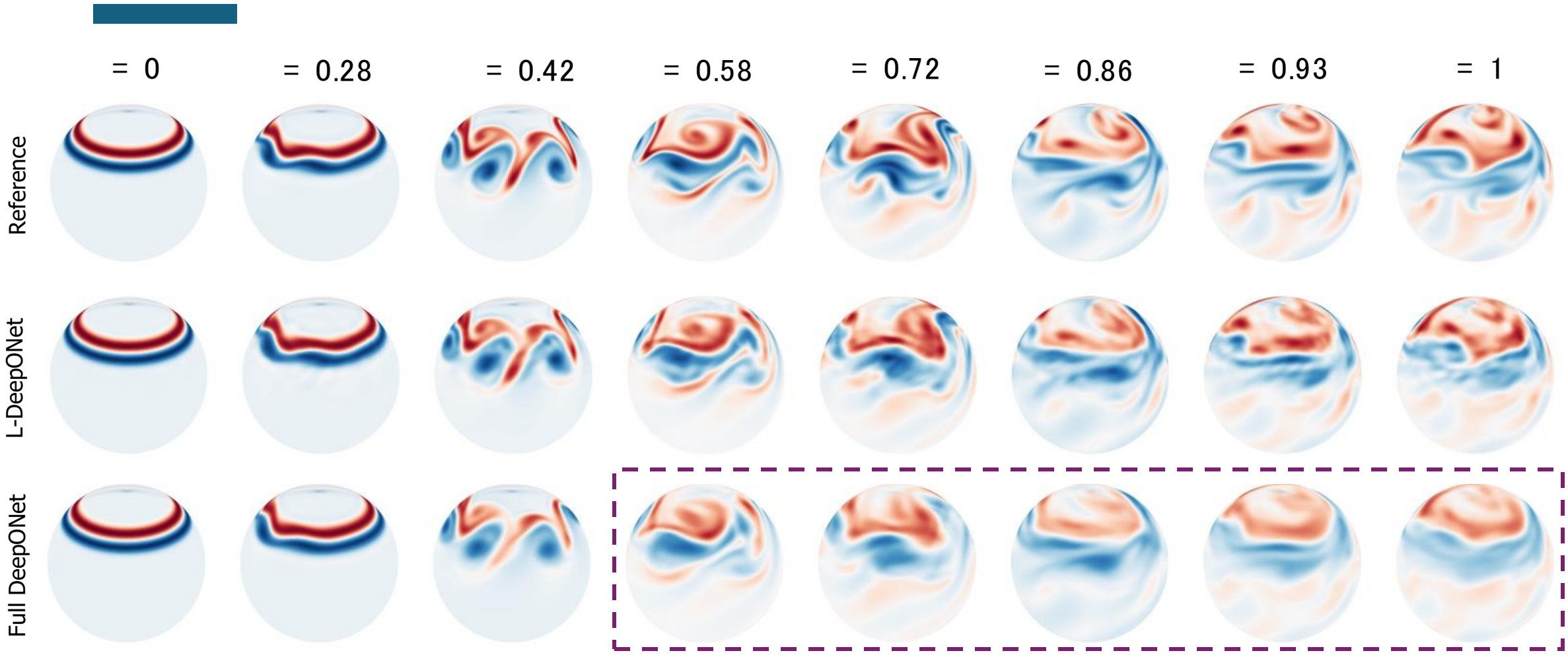




Results


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

Latent DeepONet and Full DeepONet

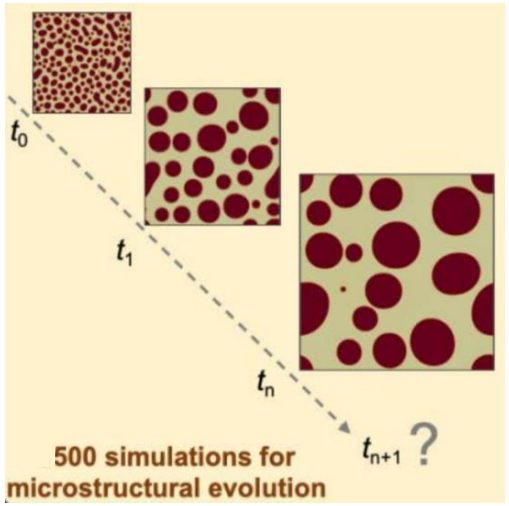


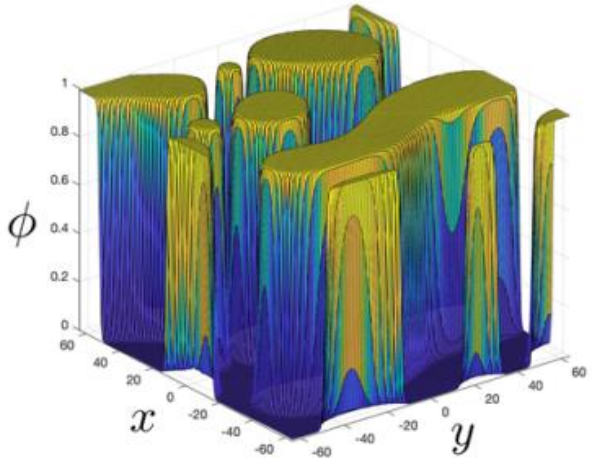
Accelerating traditional methods

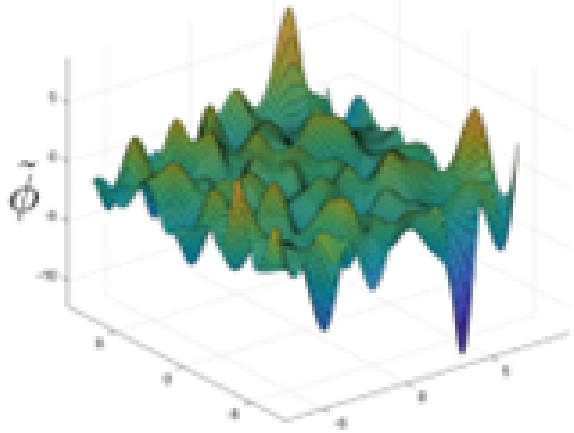
Microstructure evolution of a two-phase mixture during Spinodal decomposition

$$\frac{\partial \phi_i}{\partial t} = \nabla \cdot (M_{ij} \nabla \frac{\delta F}{\delta \phi_j})$$

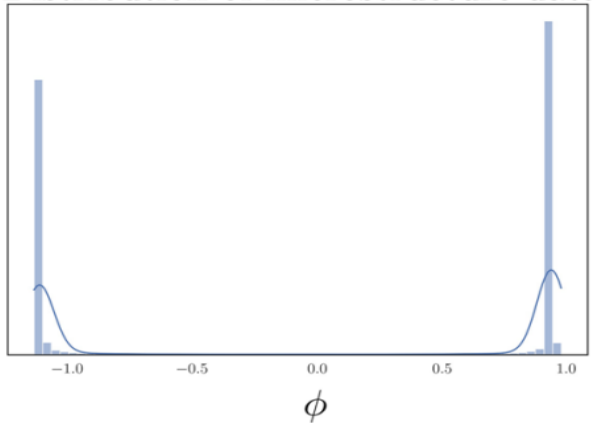
High-Fidelity Simulations



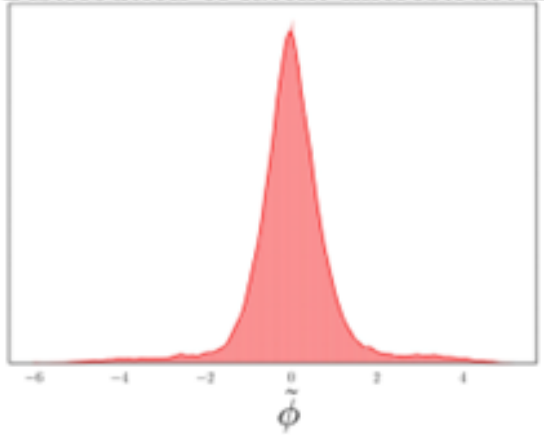




Distribution of microstructure data



Distribution of latent microstructure data



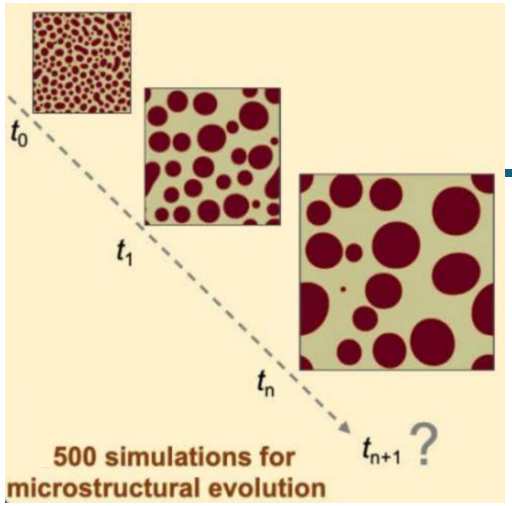
Sandia National Laboratories

Accelerating traditional methods

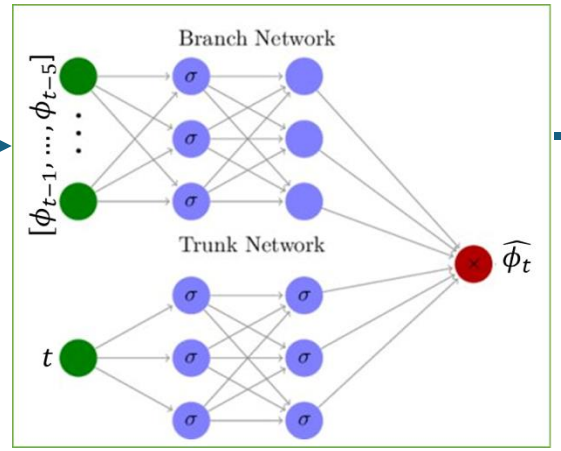
Non-linear microstructure evolution of a two-phase mixture during Spinodal decomposition

$$\frac{\partial \phi_i}{\partial t} = \nabla \cdot (M_{ij} \nabla \frac{\delta F}{\delta \phi_j})$$

High-Fidelity Simulations

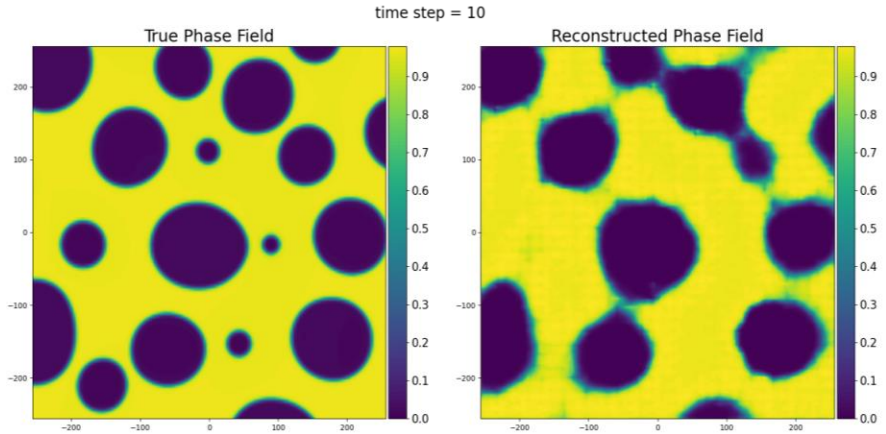


Convolutional Autoencoders (Reduced dimension = 100)



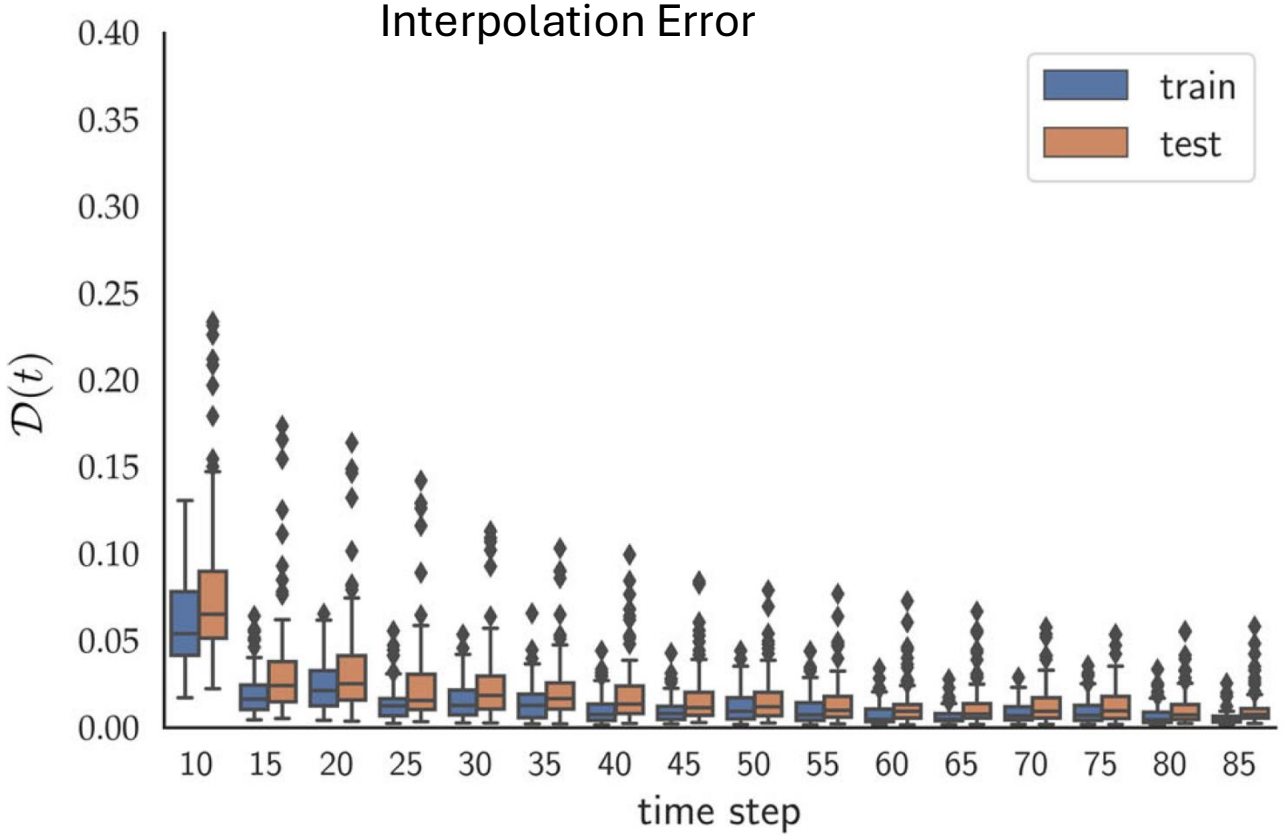
Pre-trained Transposed Convolutional decoder

ϕ_t

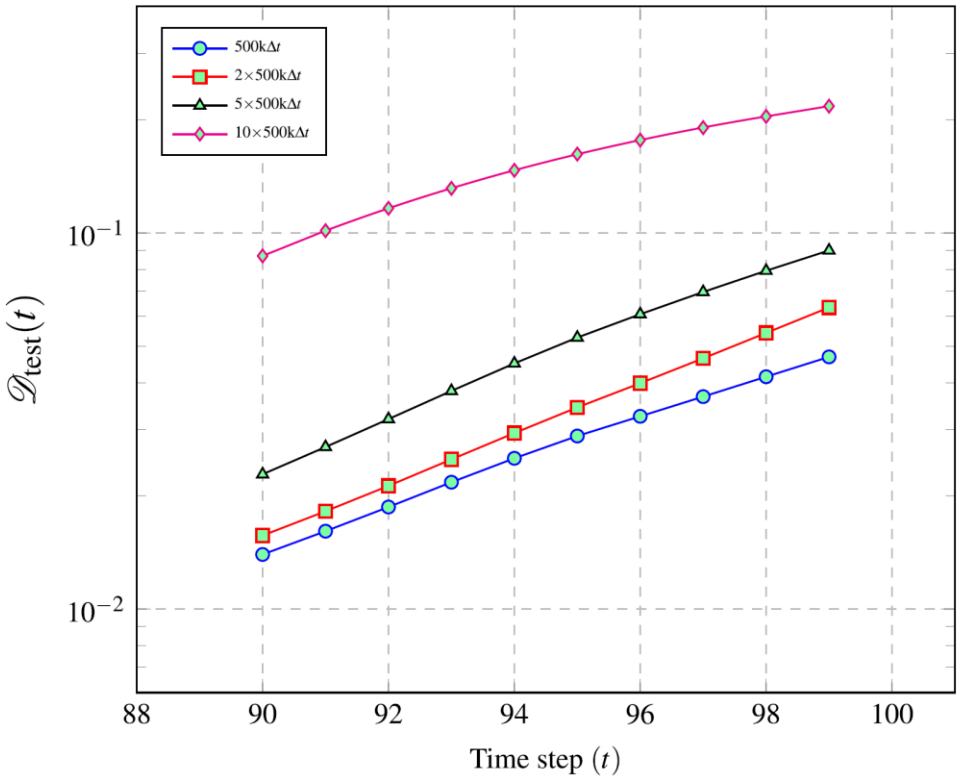


Accelerating traditional methods

Interpolation Error

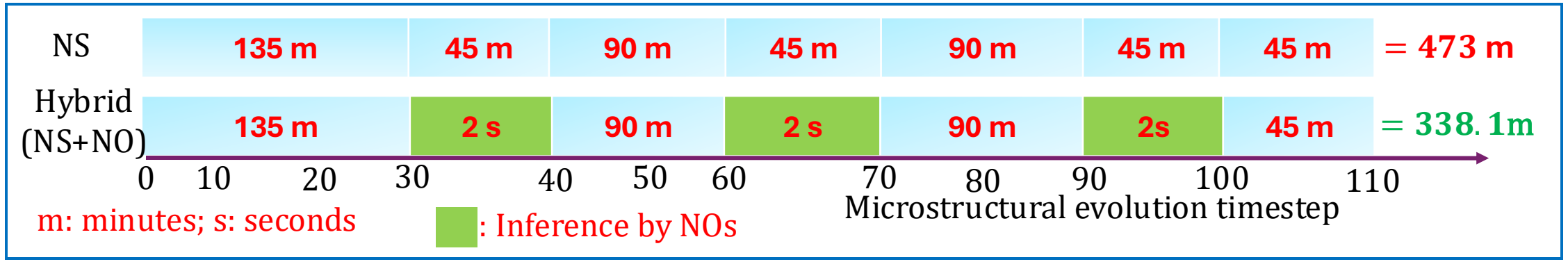


Extrapolation Error

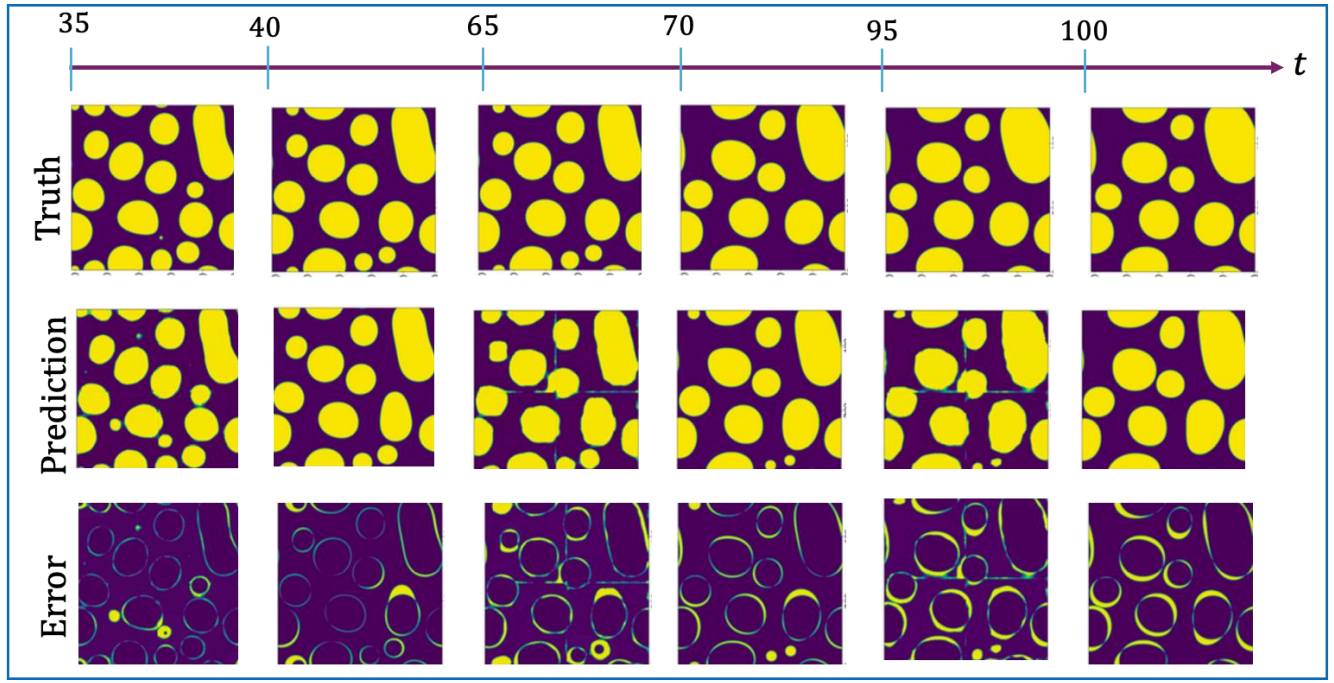


Designing a Hybrid Solver

Simulation Time



Results of the Hybrid Solver across time domain



Oommen, Vivek, Khemraj Shukla, Somdatta Goswami, Rémi Dingreville, and George Em Karniadakis. "Learning two-phase microstructure evolution using neural operators and autoencoder architectures." npj Computational Materials 8, no. 1 (2022): 190.

Can Physics be Incorporated in Latent Learning?

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



Manuscript in preparation

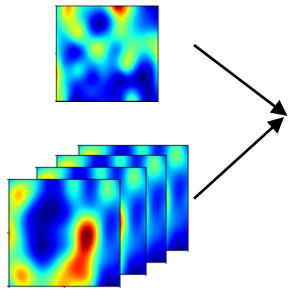
One-shot Learning: Physics Informed Latent Neural Operator

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

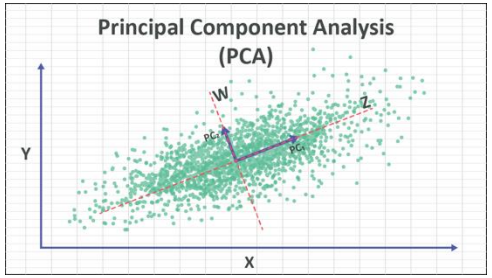
$\mathcal{G}_\theta: \mathcal{X} \to \mathcal{Y}, \quad \theta \in \Theta$

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

N(way less than data-driven)

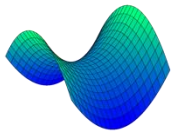


$\{\mathbf{y}_i\}_{i=1}^N$

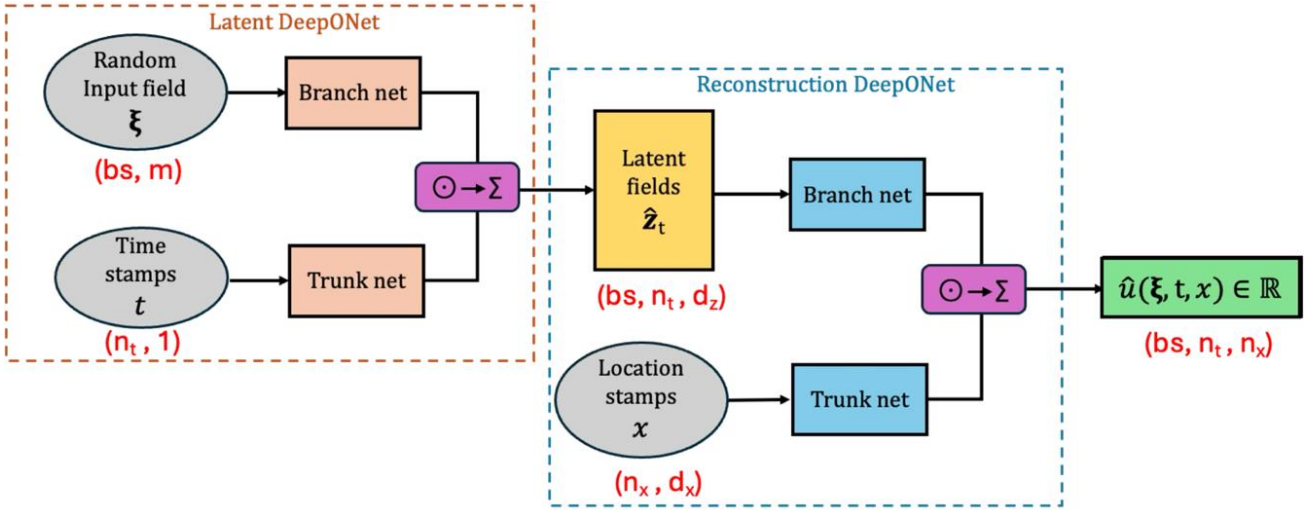


Or Autoencoder

Latent representation



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

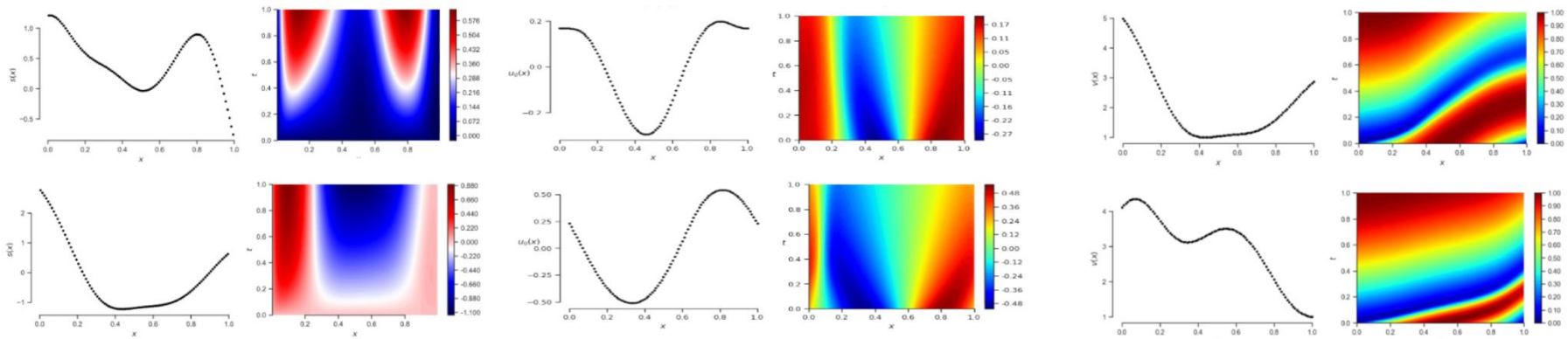


$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{data-driven}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{physics-informed}}(\boldsymbol{\theta}),$$

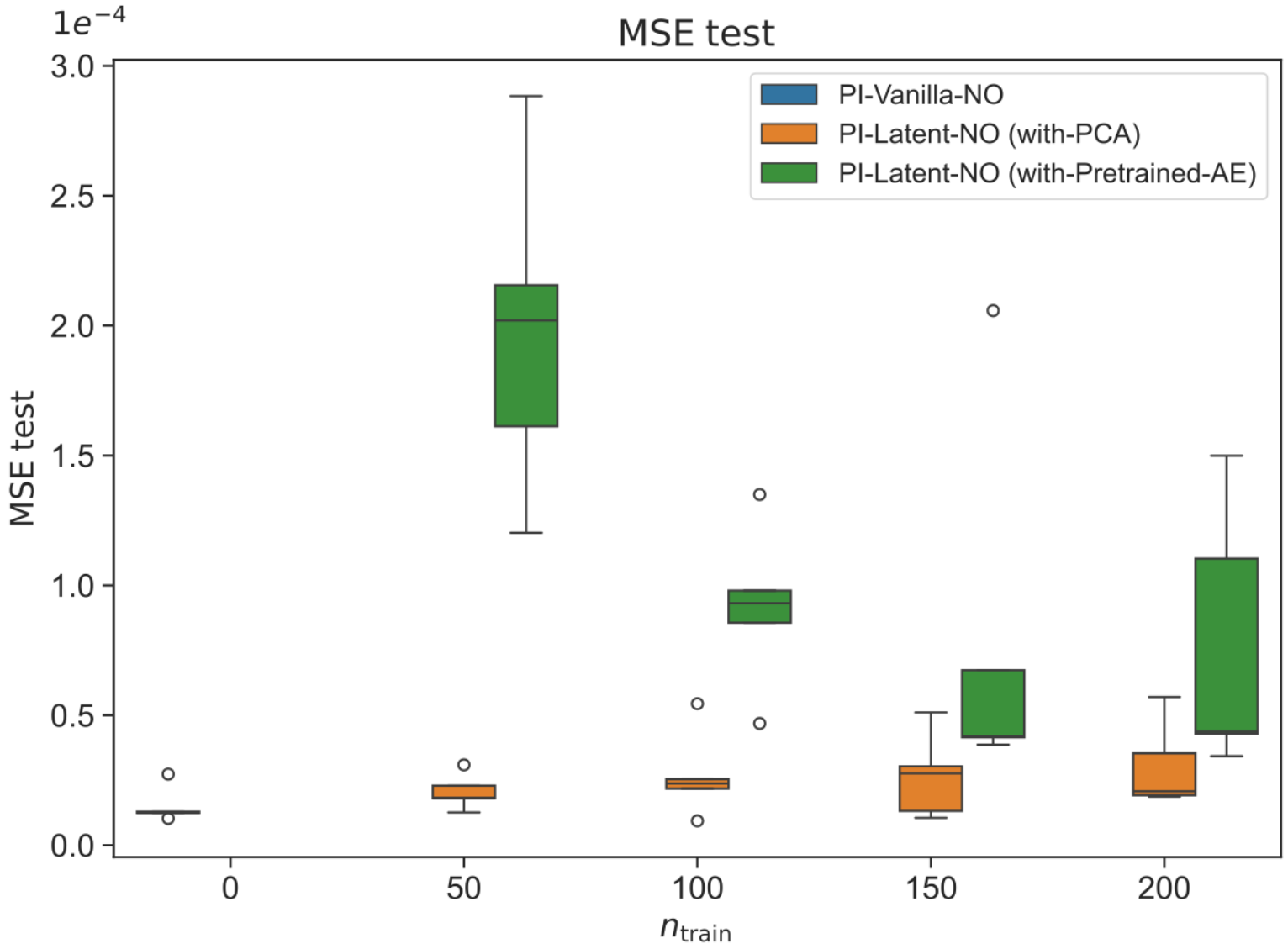
$$\mathcal{L}_{\text{data-driven}}(\boldsymbol{\theta}) = \frac{1}{n_{\text{train}}(n_t+1)} \sum_{i=1}^{n_{\text{train}}} \sum_{j=0}^{n_t} \left\| \mathbf{z}(\boldsymbol{\xi}^{(i)}, j\Delta t) - \hat{\mathbf{z}}(\boldsymbol{\xi}^{(i)}, j\Delta t) \right\|_2^2$$

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

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

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	$s(x) \sim \mathrm{GP}(0, k(x, x')),$ $\ell_x = 0.2,\ \sigma^2 = 1.0,$ $k(x, x') = \sigma^2 \exp\left\{-\frac{\|x - x'\|^2}{2\ell_x^2}\right\}.$	$g(x) \sim \mathcal{N}\left(0, 25^2\left(-\Delta + 5^2 I\right)^{-4}\right),$	$v(x) \sim \mathrm{GP}(0, k(x, x')),$ $\ell_x = 0.2,\ \sigma^2 = 1.0,$ $k(x, x') = \sigma^2 \exp\left\{-\frac{\|x - x'\|^2}{2\ell_x^2}\right\}.$
Samples			

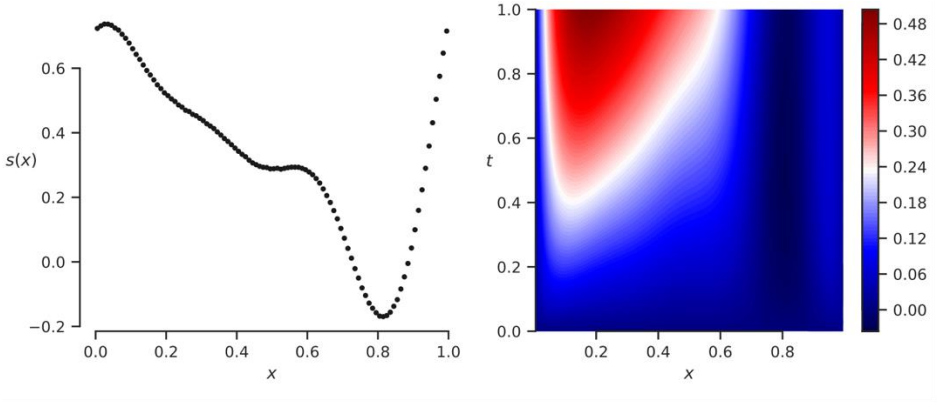
Accuracy Comparison



Reaction Diffusion Dynamics

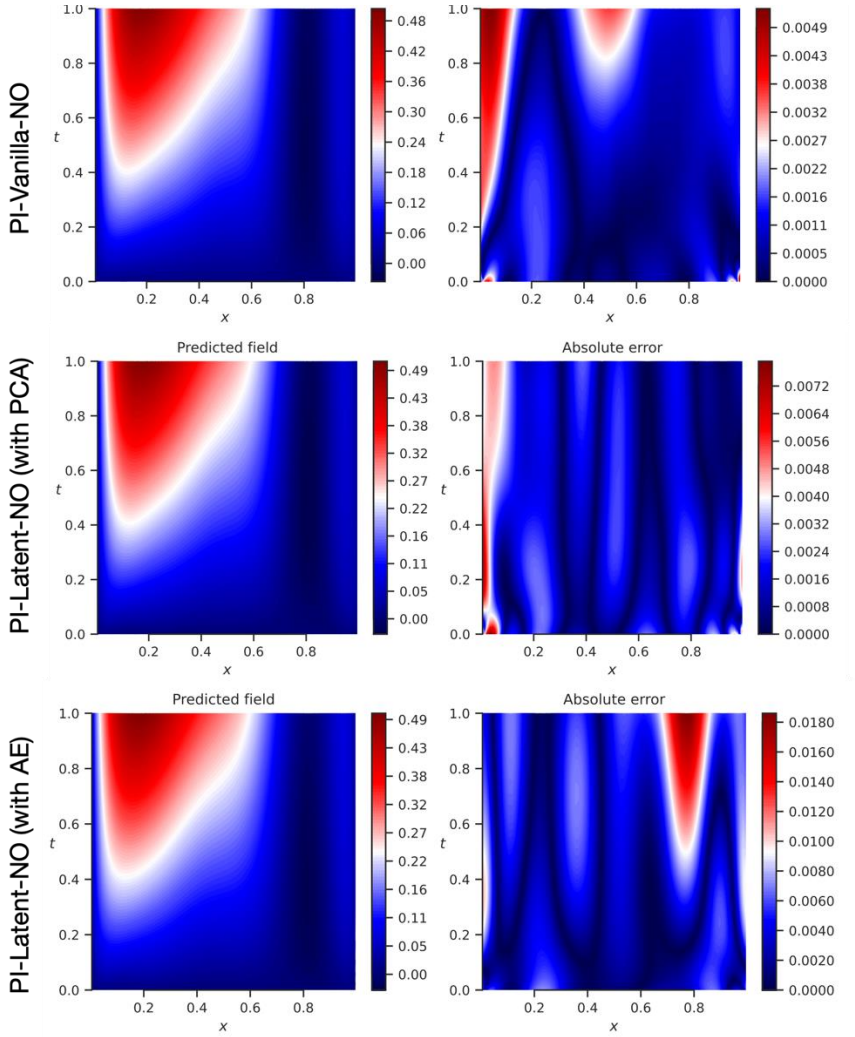
Source
(Input function)

Solution
(Ground Truth)

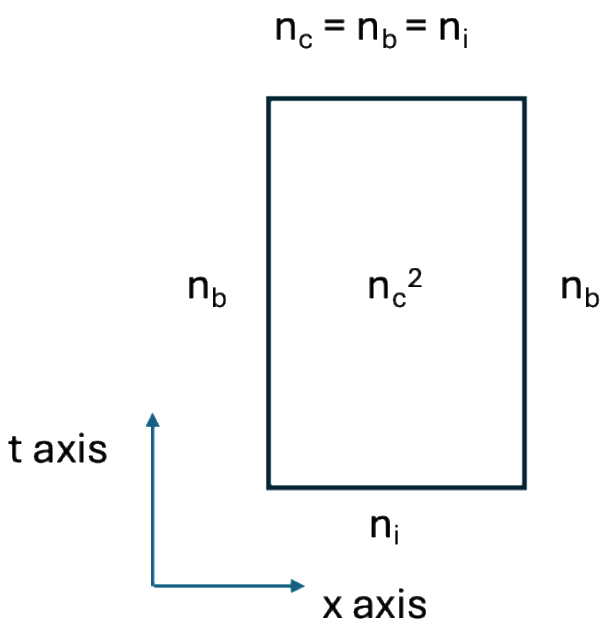


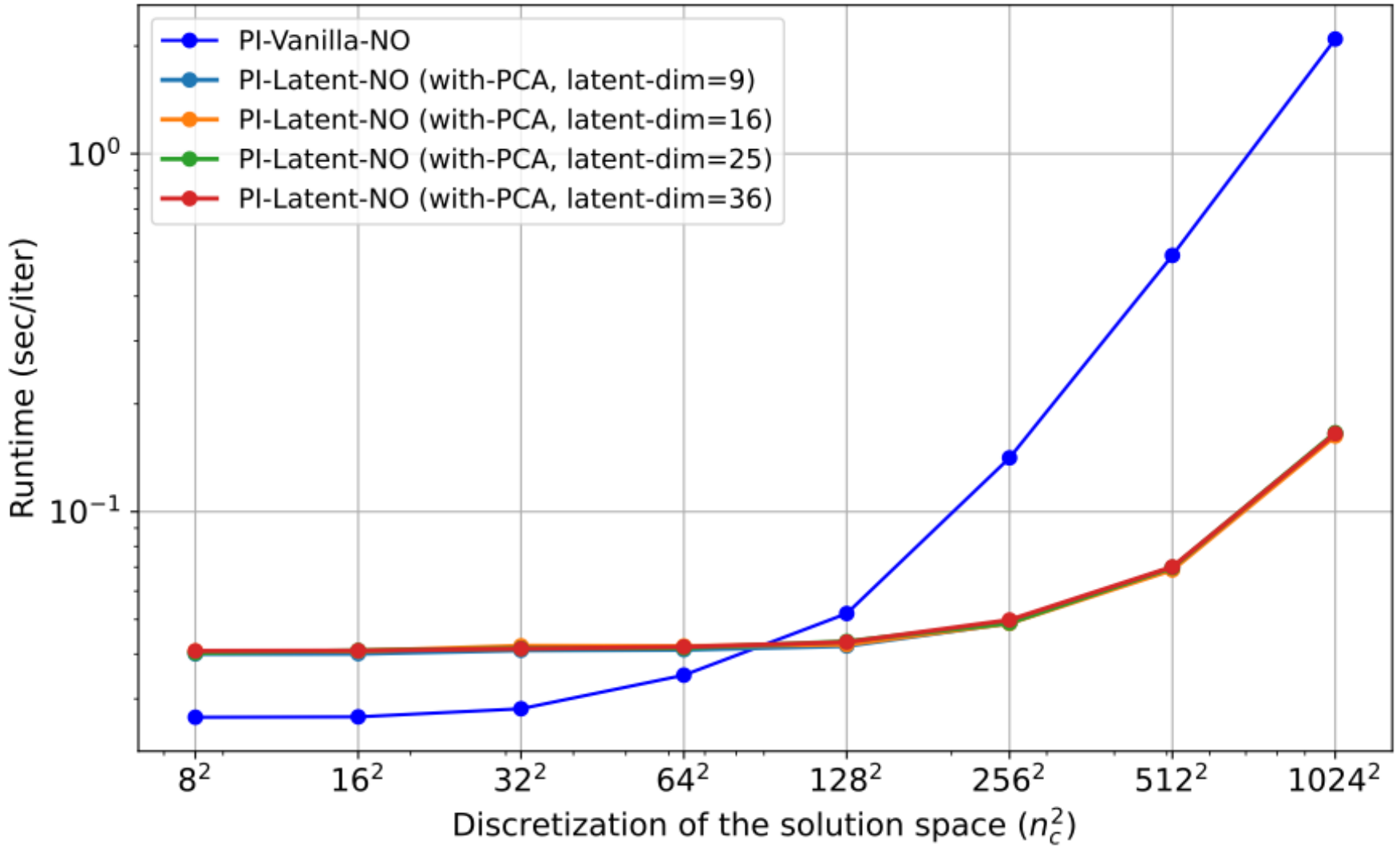
Prediction

Error

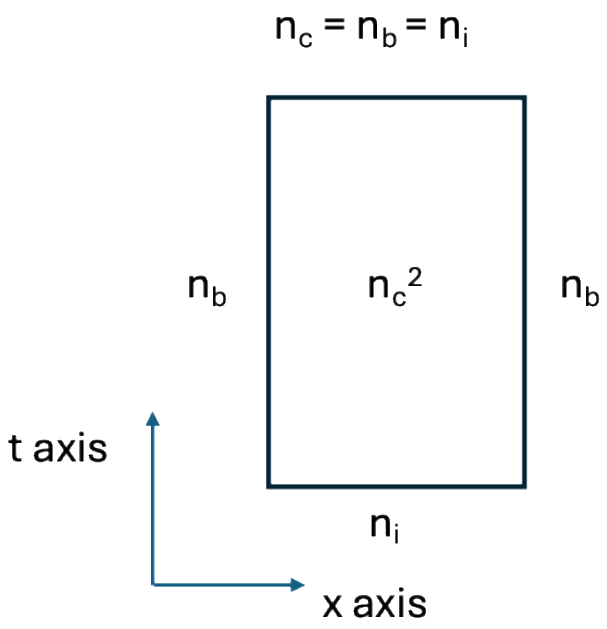


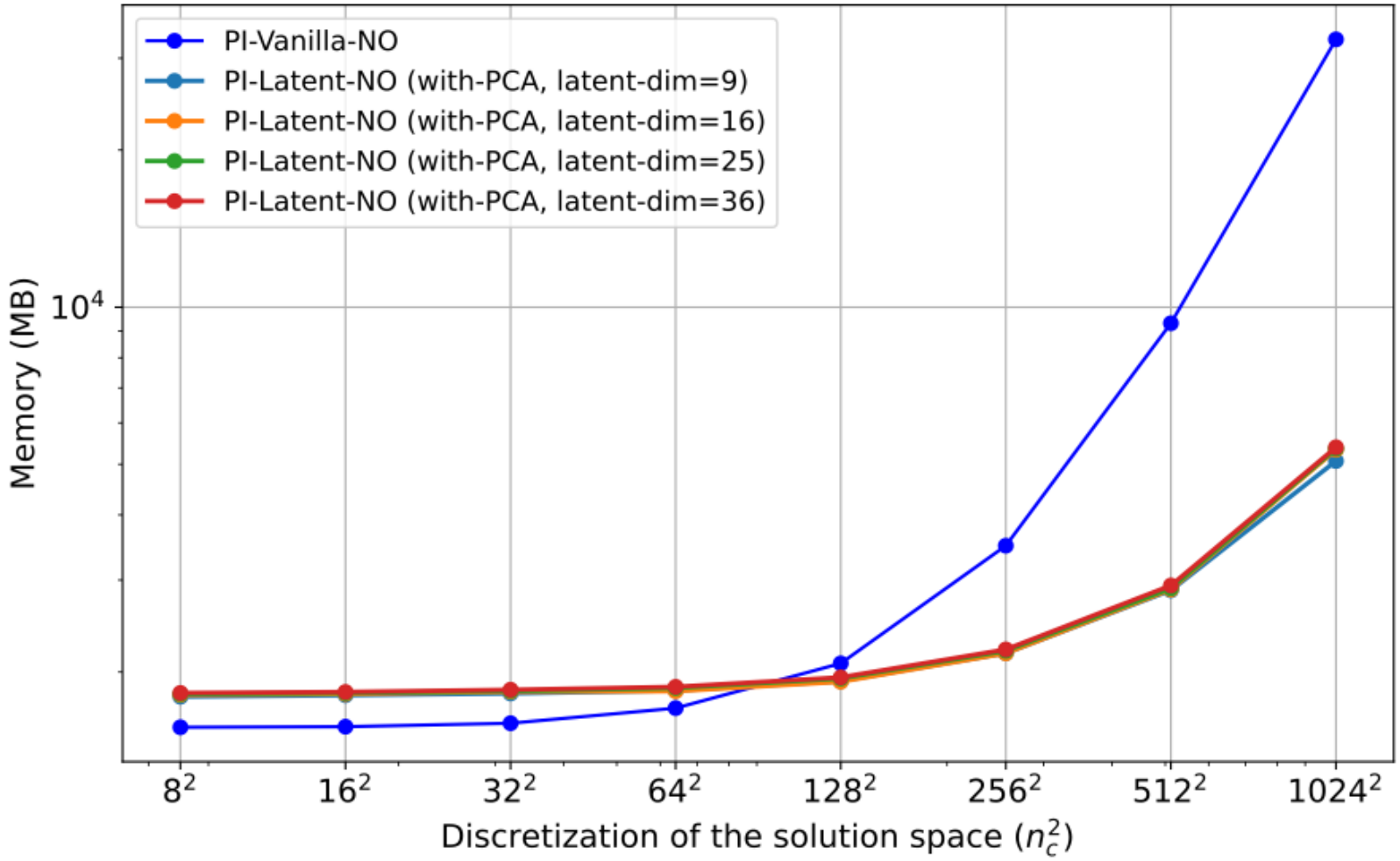
Runtime Scaling





Memory Scaling





Can Neural Operators Help Identify Systems?

Learning Hidden Physics and System Parameters with Deep Operator Networks

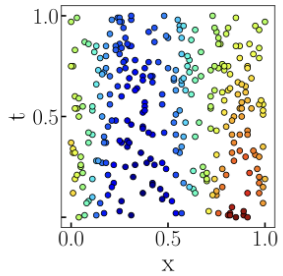


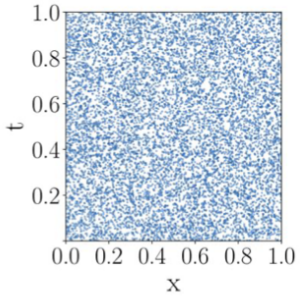

Vijay Kag, Dibakar Roy Sarkar, Birupaksha Pal, and Somdatta Goswami. "Learning Hidden Physics and System Parameters with Deep Operator Networks." *arXiv preprint arXiv:2412.05133* (2024).

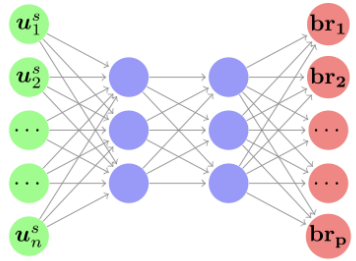
PDE : $\mathcal{N}(\mathcal{X}, u(x,t), \xi) = g$ Given: $\{x^s, t^s, u(x^s,t^s)\}$ at n locations Operator $\mathcal{G}_\theta : \mathcal{U}(x^s,t^s) \to u(x,t)$ Operator $\mathcal{I}_\phi : \mathcal{U}(x^s,t^s) \to \xi$

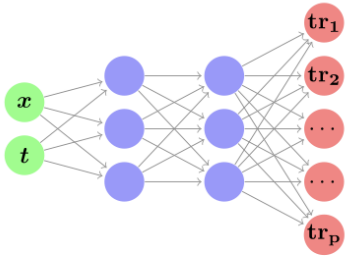
Step 1: Training solution operator

Full-field reconstruction using sensor data



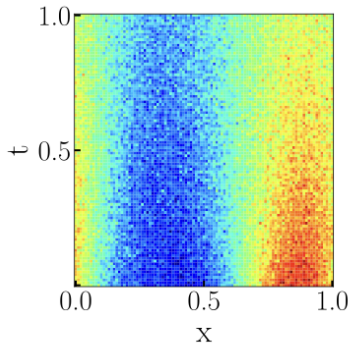






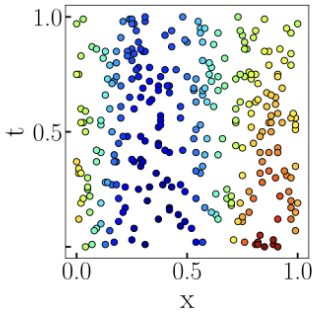
$$\mathcal{G}_\theta(u(x^s,t^s))(x,t)$$

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}[\mathcal{G}_\theta(u_i(x^s,t^s))(x^s,t^s) - u_i(x^s,t^s)]^2 \to \boldsymbol{\theta}^*$$



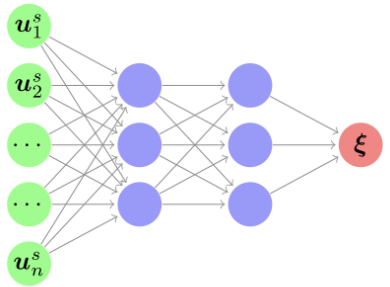
Step 2: Learn parameter operator

Unknown parameter learning network



$\mathcal{G}_\theta(u(x^s,t^s))(x,t)$

initialize with $\theta = \boldsymbol{\theta}^*$

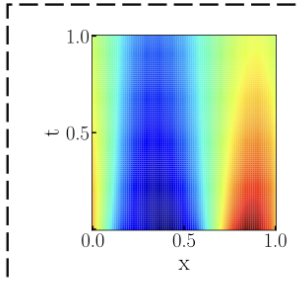


continual learning of the reconstruction network

$$\mathcal{L}(\theta,\phi) = \mathcal{L}^{\text{pde}}(\theta,\phi)(x,t) + \mathcal{L}^{\text{data}}(\theta)(x^s,t^s)$$

$\mathcal{I}_\phi(u(x^s,t^s))$

$\boldsymbol{\theta}^*, \boldsymbol{\phi}^*$



, $\boldsymbol{\xi}$

Finding Viscosity from Sensor Measurements

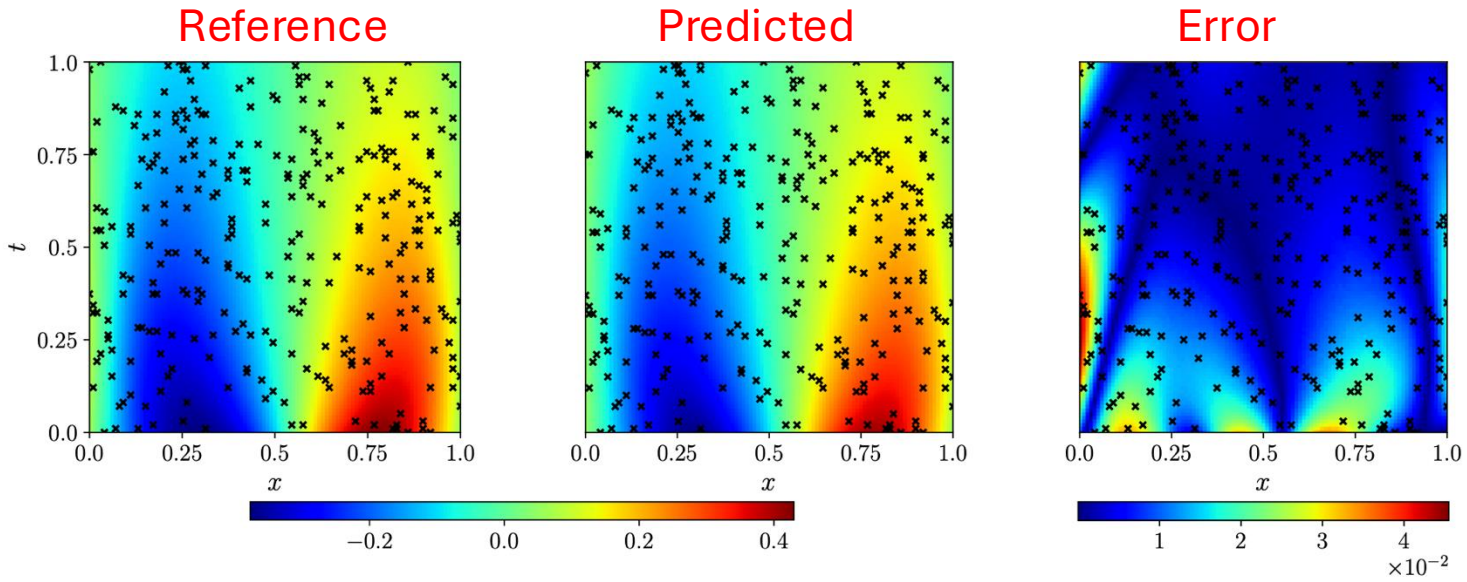
$$\frac{\partial s(x,t)}{\partial t} + s\frac{\partial s(x,t)}{\partial x} - \nu\frac{\partial^2 s(x,t)}{\partial x^2} = 0,$$

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

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

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

Given: s at 300 locations marked with x

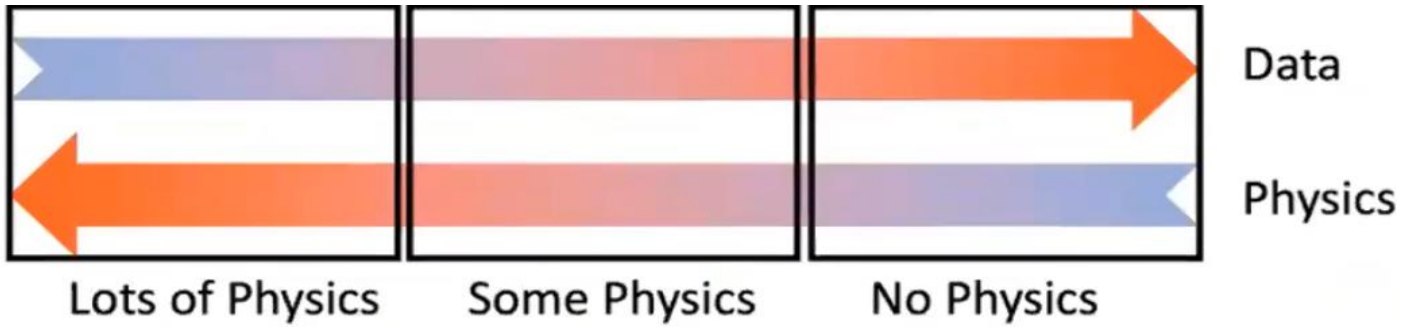


$\nu_{true} = 0.032$

$\nu_{pred} = 0.032$

Reconstruction Error: $L_2 \rightarrow 0.6\%$

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









Thank you!