Towards Foundation Models for Scientific Machine Learning

Presented by **Amir Gholami**

University of California Berkeley, ICSI

In collaboration with:

S. Subramanian, W Chen, S. Zhe, M. Kirby, K. Keutzer, M. Mahoney

Berkeley
UNIVERSITY OF CALIFORNIA

BAIR
BERKELEY ARTIFICIAL INTELLIGENCE RESEARCH

Outline

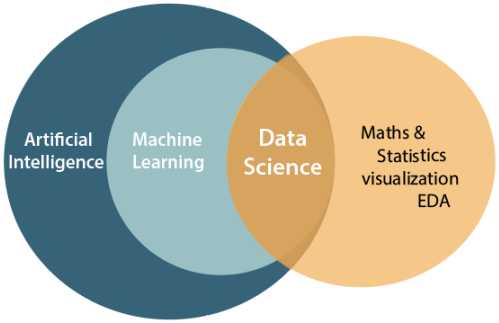
- **Foundation Models for SciML?**
- Incorporating Physical Laws into Learning
- Conclusions and Future Work

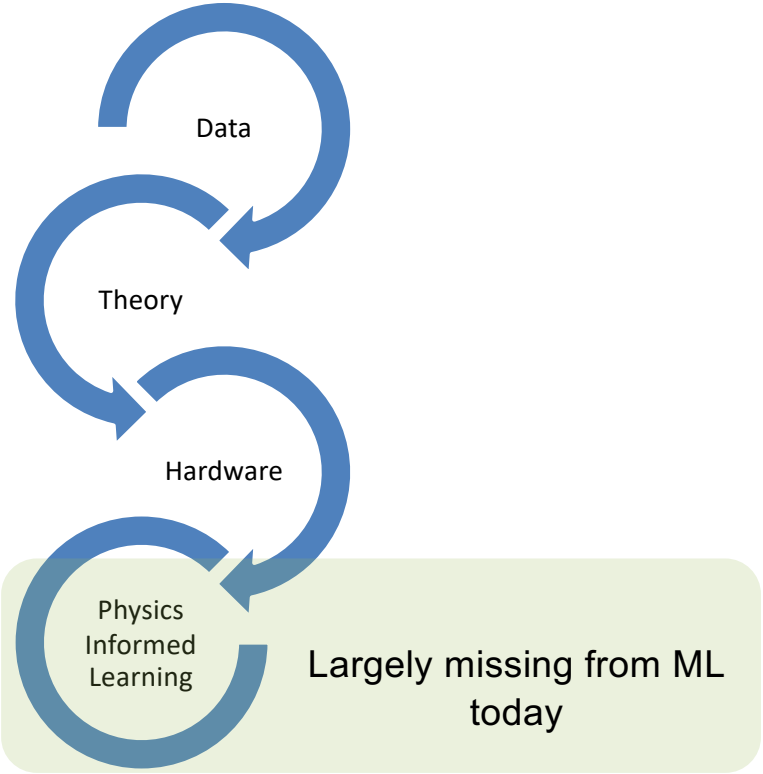
Physics Informed Learning

Computational Science is an important tool that we can use to incorporate physical invariances into learning, but until recently it was missing from mainstream ML.

"**Computational Science** can **analyze past events** and **look into the future**. It can explore the effects of thousands of scenarios for or in lieu of actual experiment and be used to study events beyond the reach of expanding the boundaries of experimental science"
–Tinsley Oden, 2013

To make further progress in ML it is crucial that we incorporate computational science into learning.





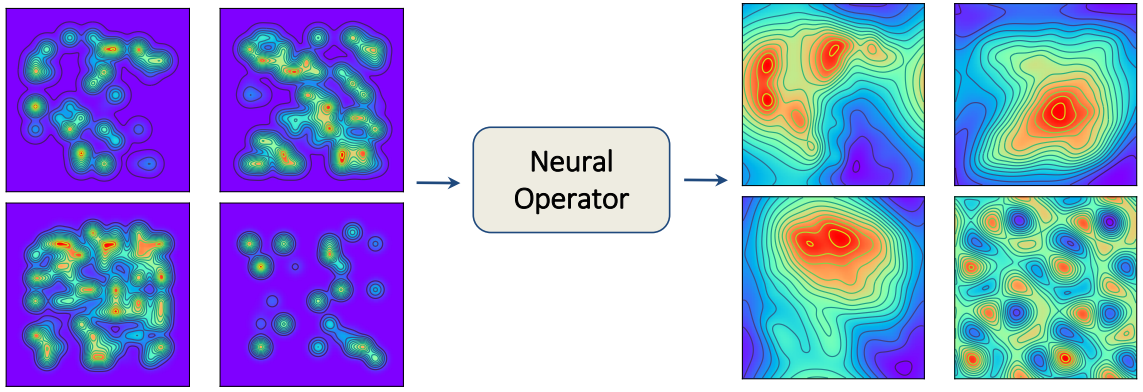
Dr. J. Tinsley Oden's Commemorative Speech: "THE THIRD PILLAR: The Computational Revolution of Science and Engineering", Honda Prize, 2013.

Foundation Models for SciML Tasks?

Create and pre-train on diverse PDE systems

Vary/Sample all inputs (PDE coefficients, source functions, …)
Include multiple differential operators, predict PDE solution

$$\nabla \cdot \mathrm{K} \nabla u + v \cdot \nabla u + \ \ldots = f$$

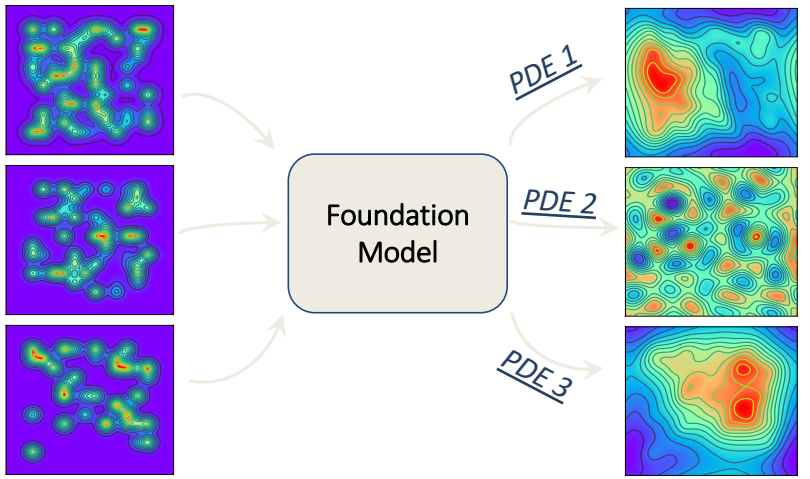


Inputs
f, K, v, …

Outputs
u

Foundation Models for SciML

Solve multiple systems using the same pre-trained model, outperforming training from scratch



Inputs
f, K, v, …

Outputs
u

Methods for Incorporating Physics into Learning

- Train on large amount of data (and hope) that the NN learns the constraints

Obtain/Simulate a lot of data

Train the NN on this dataset

Q: If we scale the NN/Data, can the model be used on out of distribution examples?

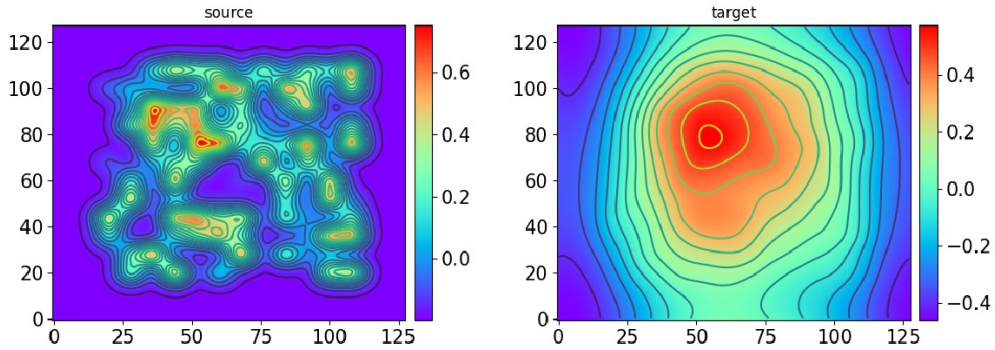
Main problems:

- No guarantee that the model obeys the physics laws (e.g. conservation laws)
- May require a lot of training data and obtaining/simulating these data is not always feasible

[Lu et al., 2019; Bhattacharya et al., 2020; Nelsen & Stuart, 2020; Li et al. 2020, Patel et al., 2021]

PDE system 1: Poisson's equation

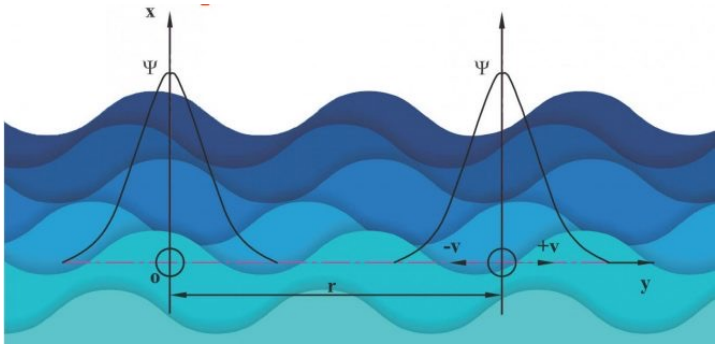
$$-\mathrm{div} K \nabla u = f \quad \text{in } \Omega$$



- Inputs: diffusion tensor (K), forcing/source function (f)
- Outputs: solution to the PDE (u)
- Training dataset: sample diffusion tensors and source functions from a training distribution

PDE System 2: Advection Diffusion

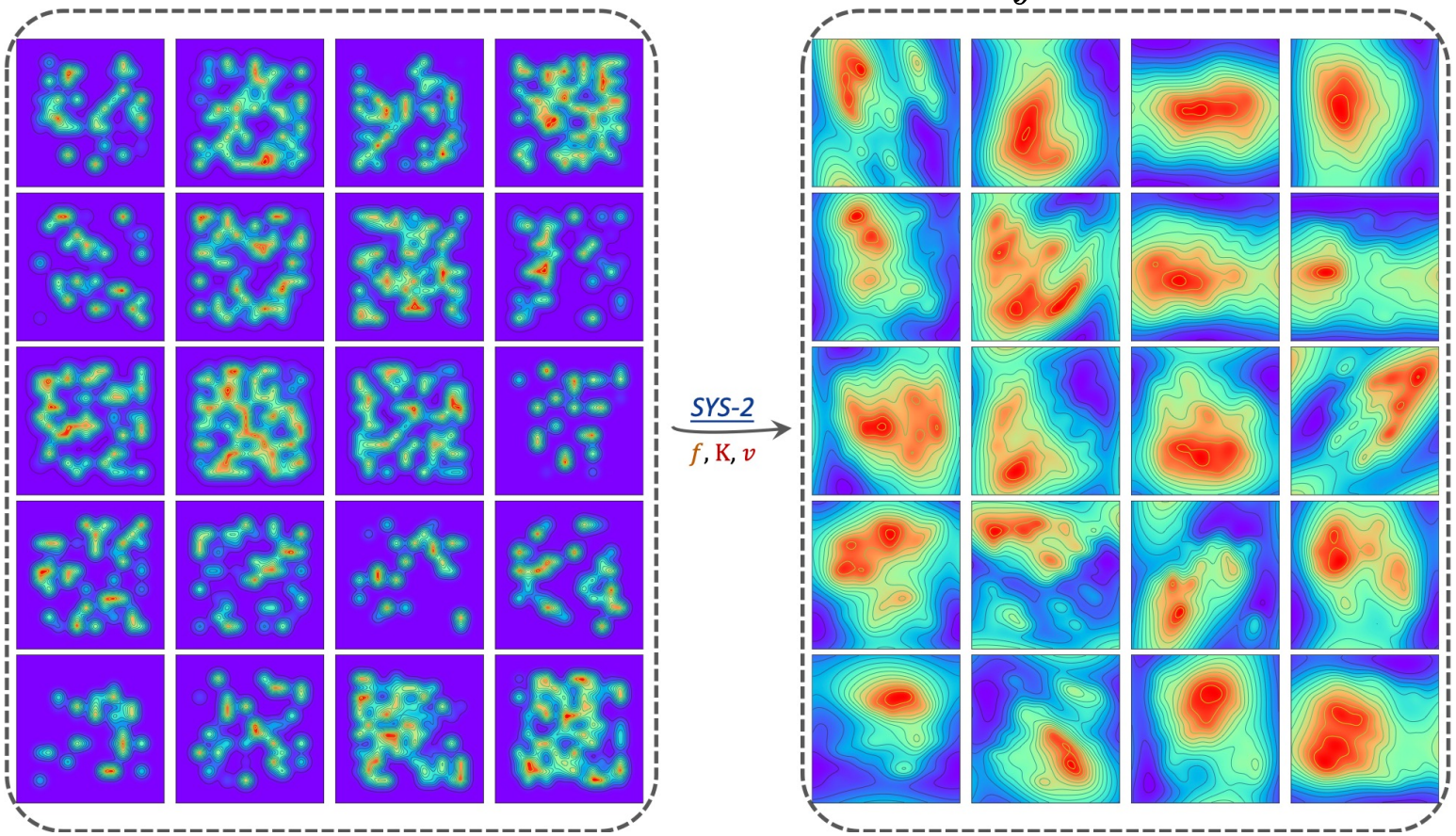
$$-\mathrm{div} K \nabla u + v \nabla u = f \quad \text{in } \Omega$$



- Inputs: diffusion tensor (K), wave speed (v), forcing/source function (f)
- Outputs: solution to the PDE (u)
- Training dataset: sample diffusion tensors, wave speeds, and source functions from a training distribution

Training Dataset

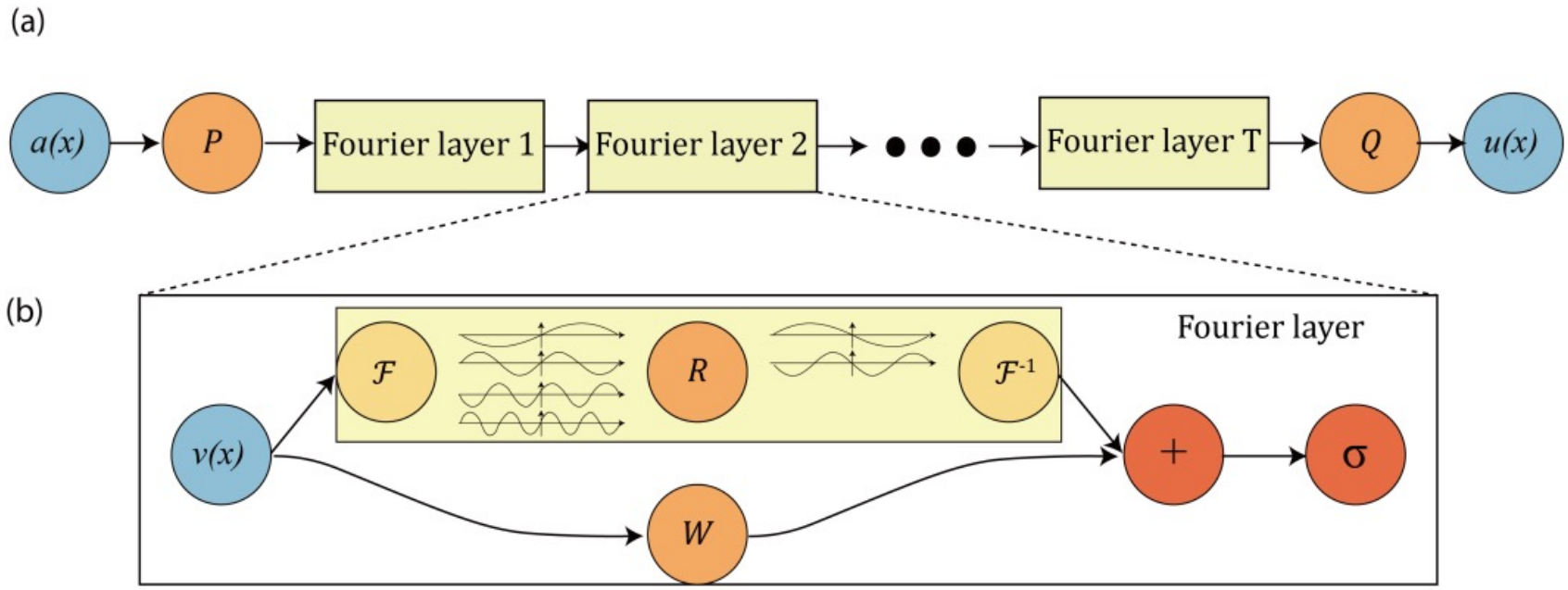
$$-\mathrm{div}K\nabla u + v\nabla u = f$$



Foundation Model for SciML Tasks?

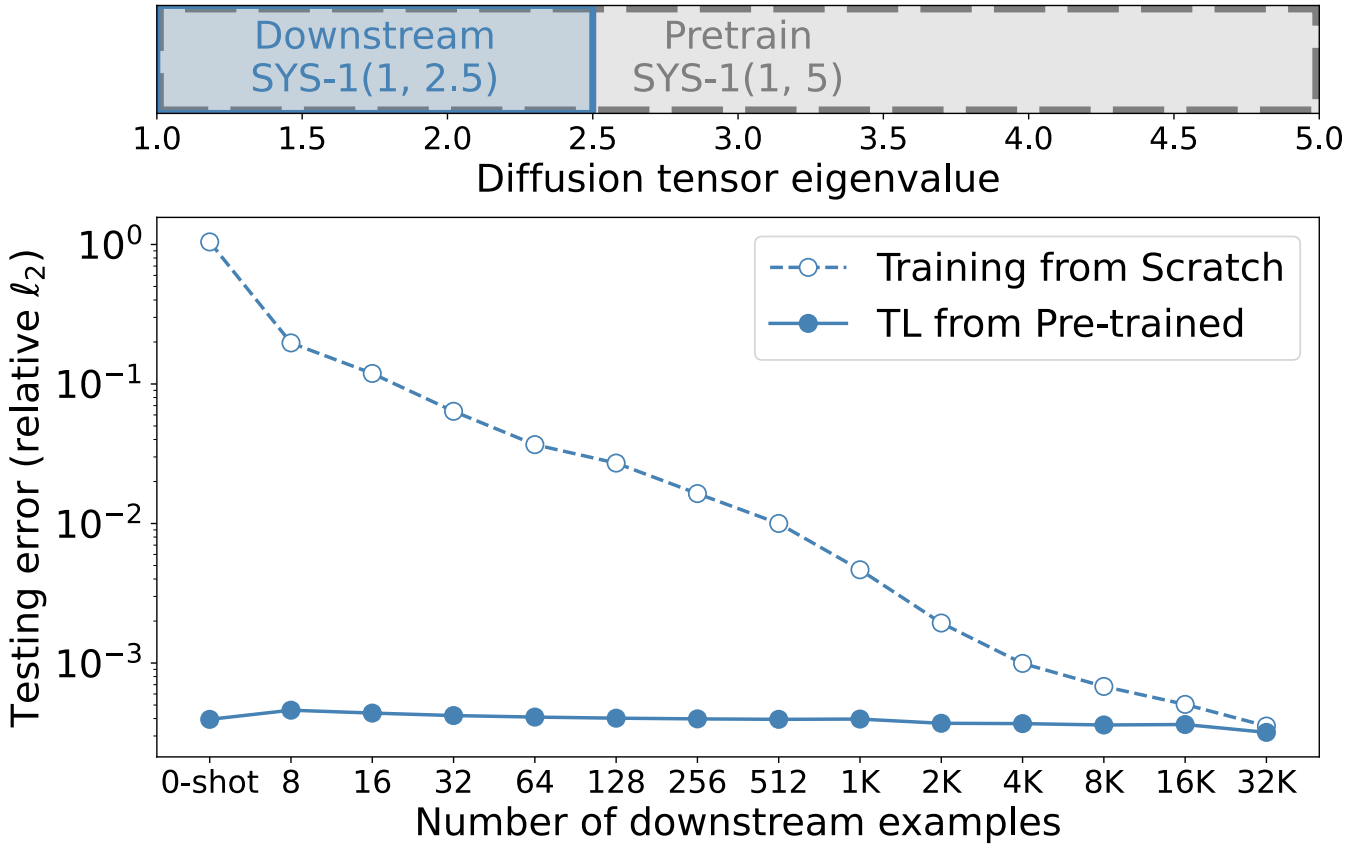
- Choose FNO as the baseline model for all experiments
- Two common PDE systems: Poisson and Advection-Diffusion
- General strategy:
  - Create a base, ``large'' dataset to pretrain an FNO
  - Test pretrained vs randomly initialized FNO on new ``downstream'' dataset
- Control Knobs:
  - **Data scale**: change amount of downstream dataset availability
  - **Model scale**: Scale model parameters to understand scaling laws
  - **Downstream Task**: change ``physical distance'' of downstream dataset (OOD)

FNO is a based on a transformer (mixing) architecture



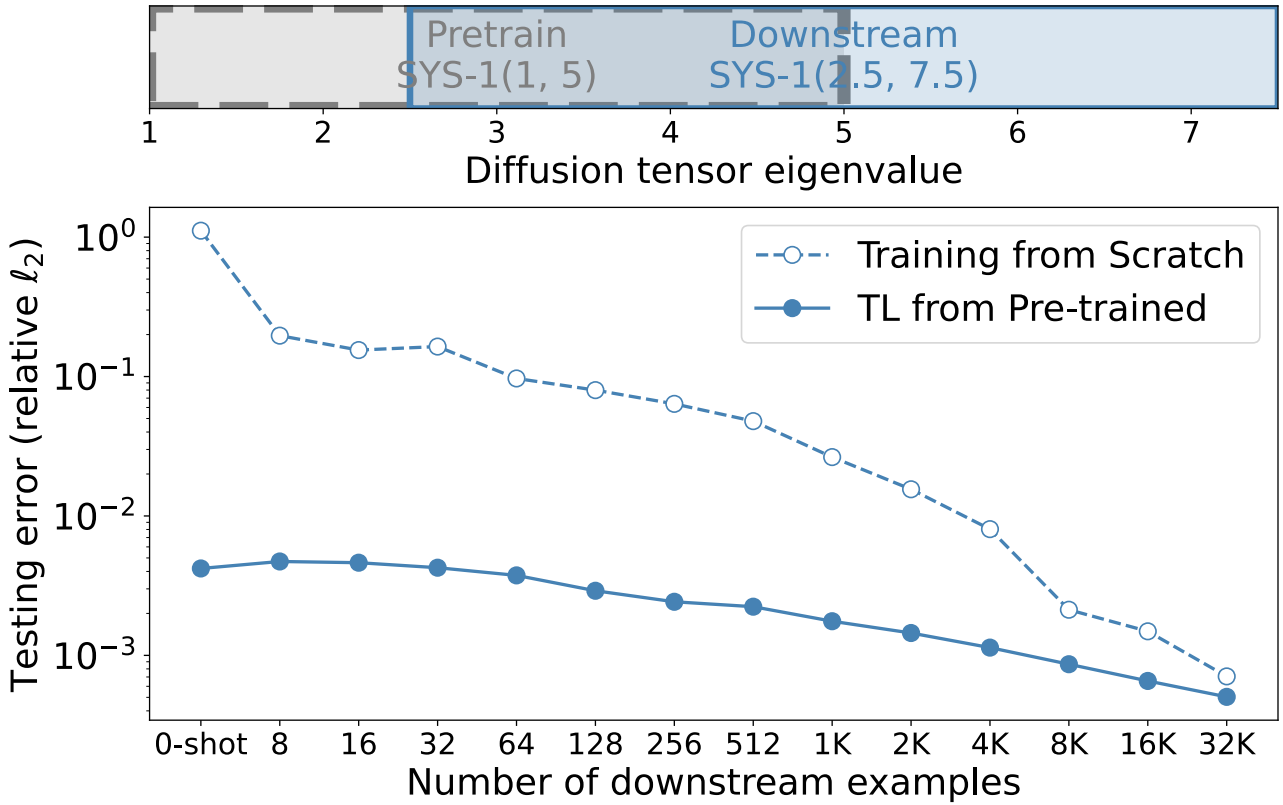
Transfer Learning

$$-\mathrm{div} K \nabla u = f \quad \text{in } \Omega$$



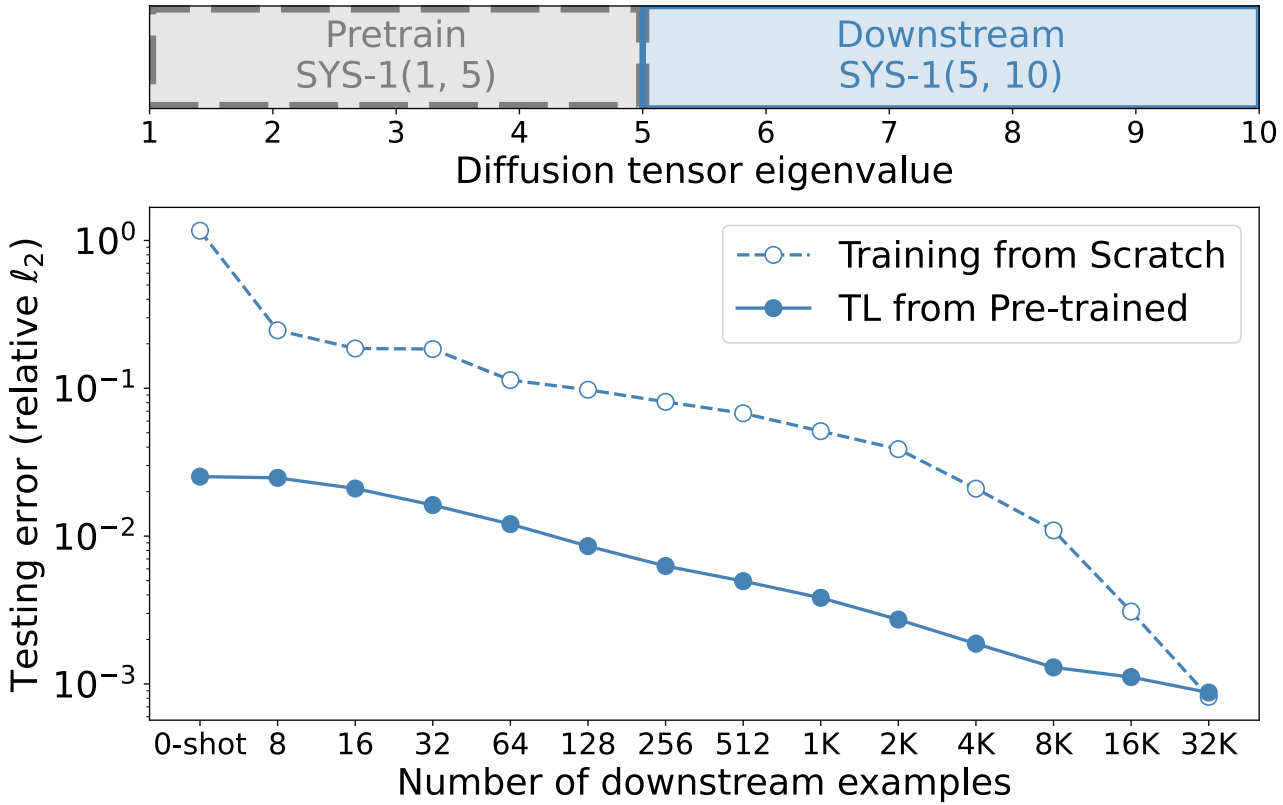
Out Of Distribution Transfer Learning

$$-\mathrm{div} K \nabla u = f \quad \text{in } \Omega$$



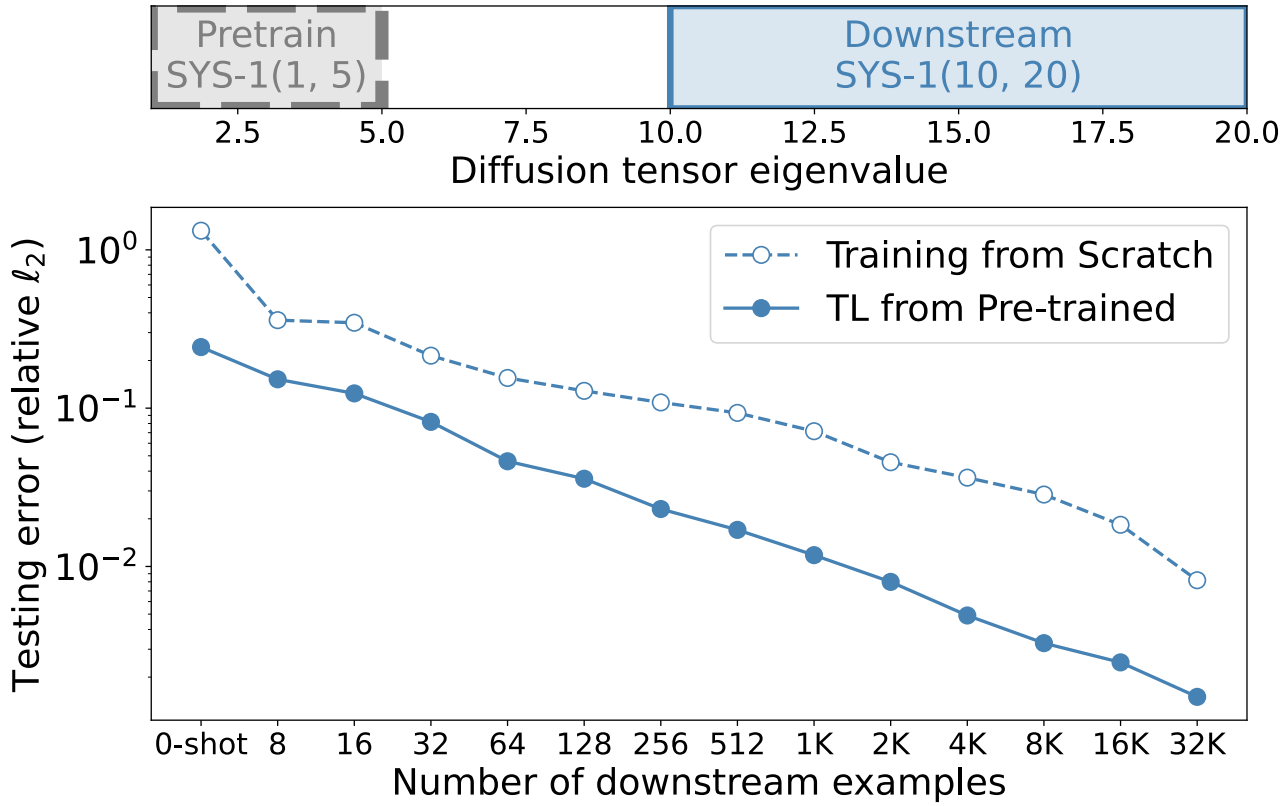
Out Of Distribution Transfer Learning

$$-\mathrm{div} K \nabla u = f \quad \text{in } \Omega$$

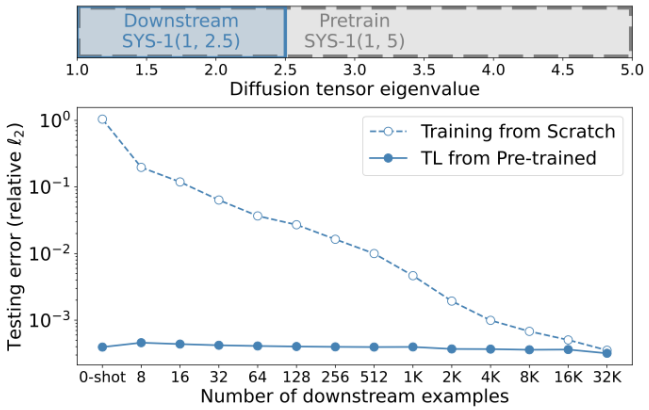


Out Of Distribution Transfer Learning

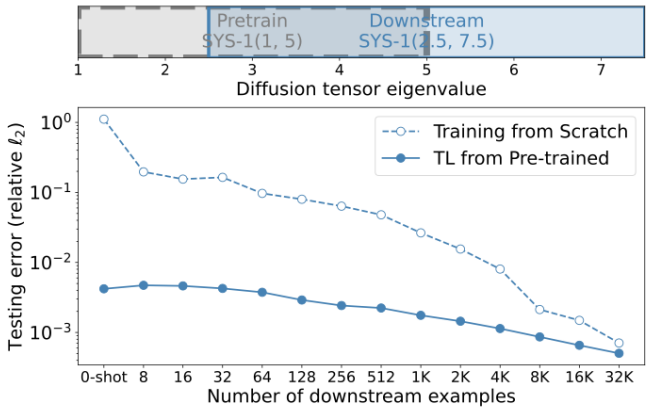
$$-\mathrm{div} K \nabla u = f \quad \text{in } \Omega$$



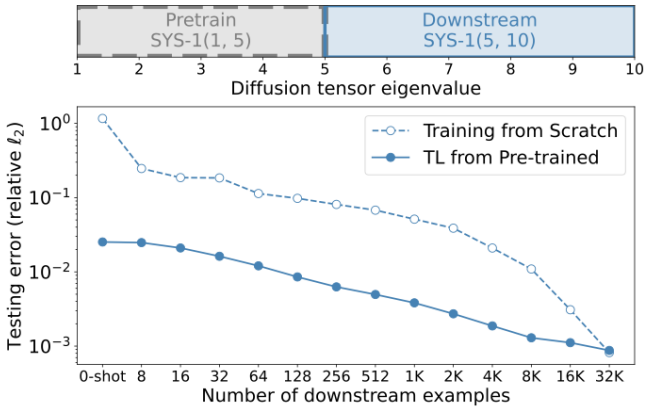
Out Of Distribution Transfer Learning



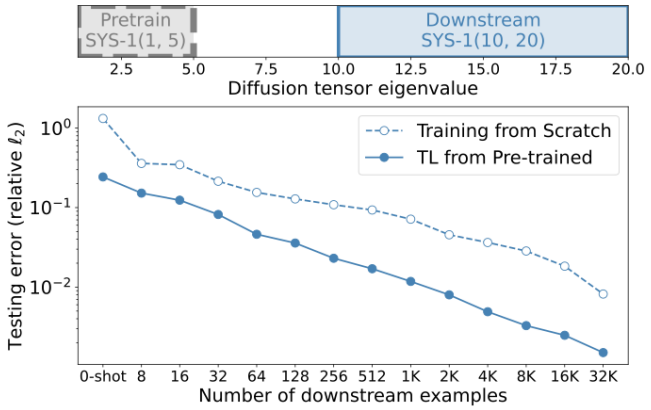
(a) *SYS-1(1,2.5)*



(b) *SYS-1(2.5,7.5)*

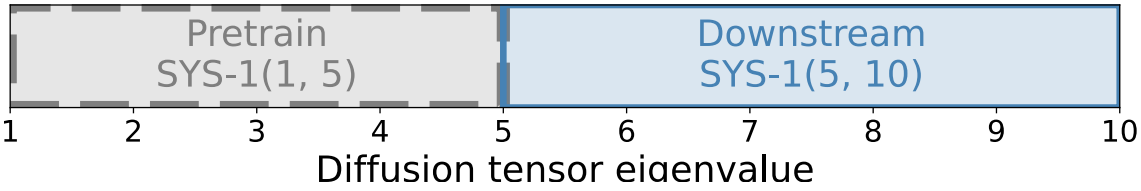


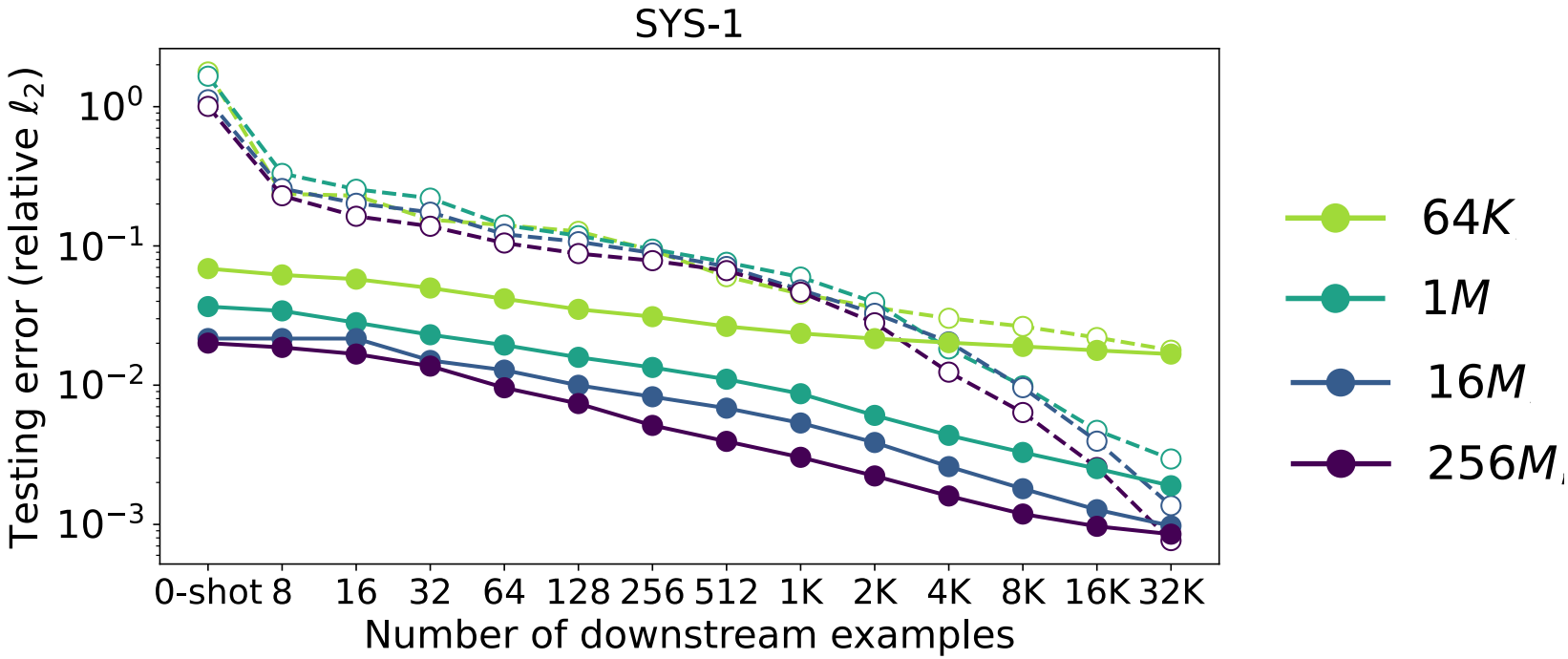
(c) *SYS-1(5,10)*



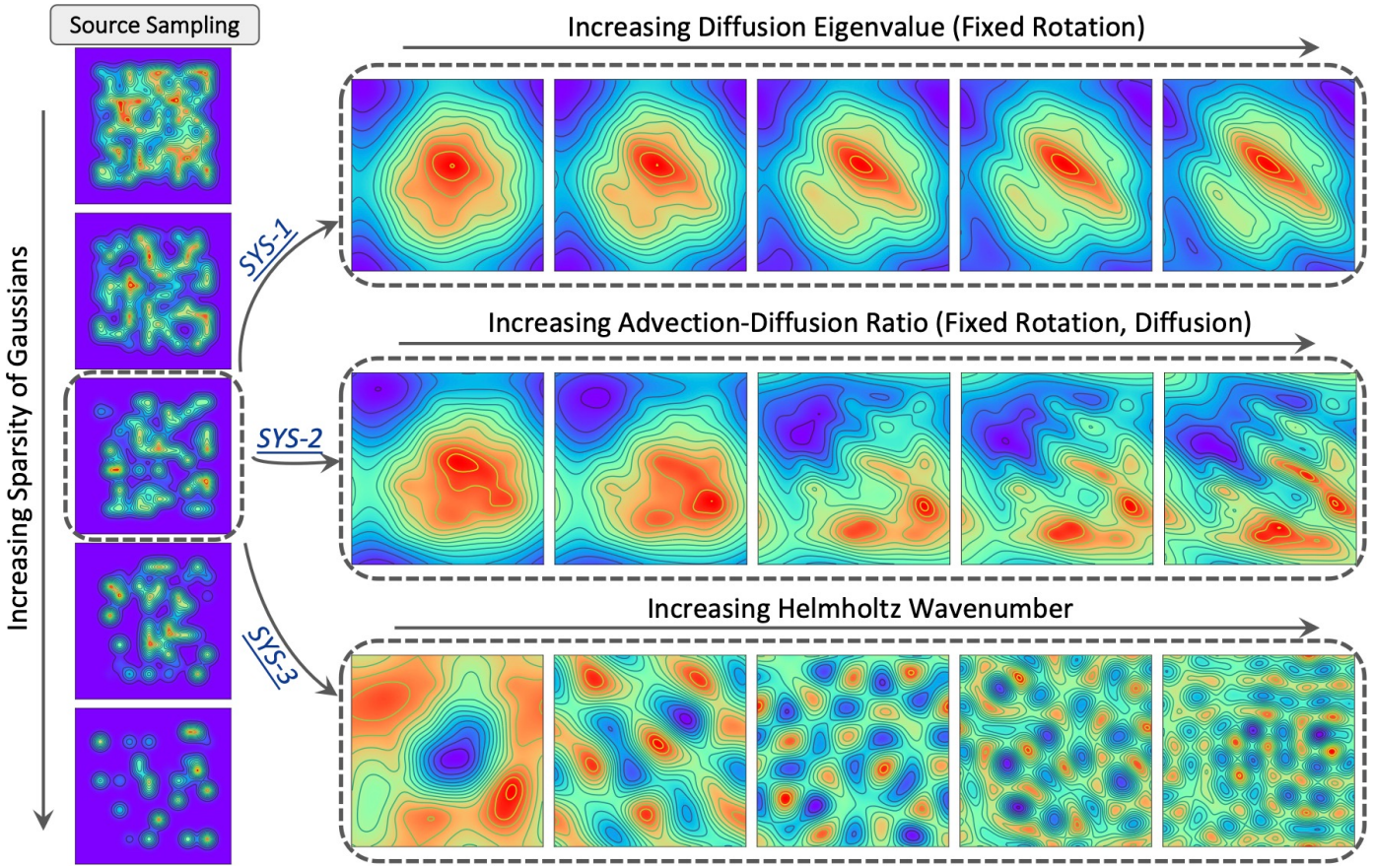
(d) *SYS-1(10,20)*

Transfer Learning Behavior vs Model Size



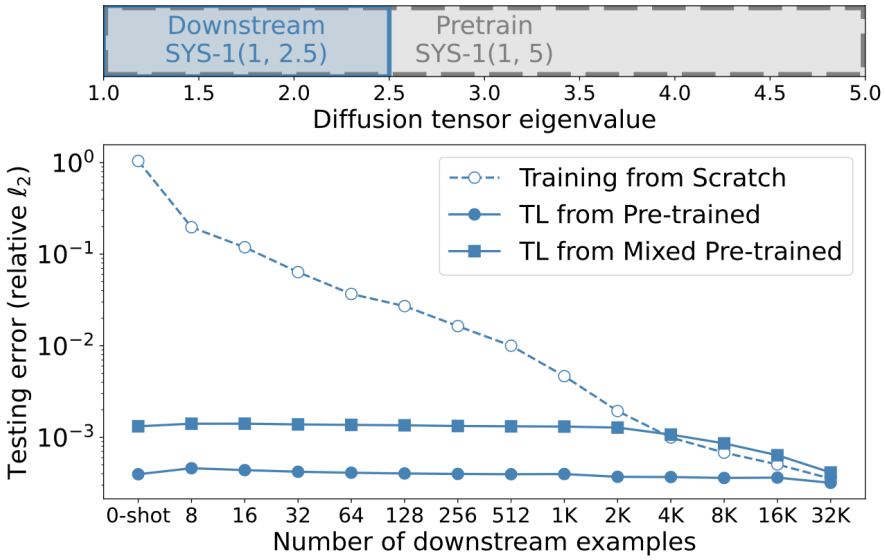


Pre Train on Multiple Systems



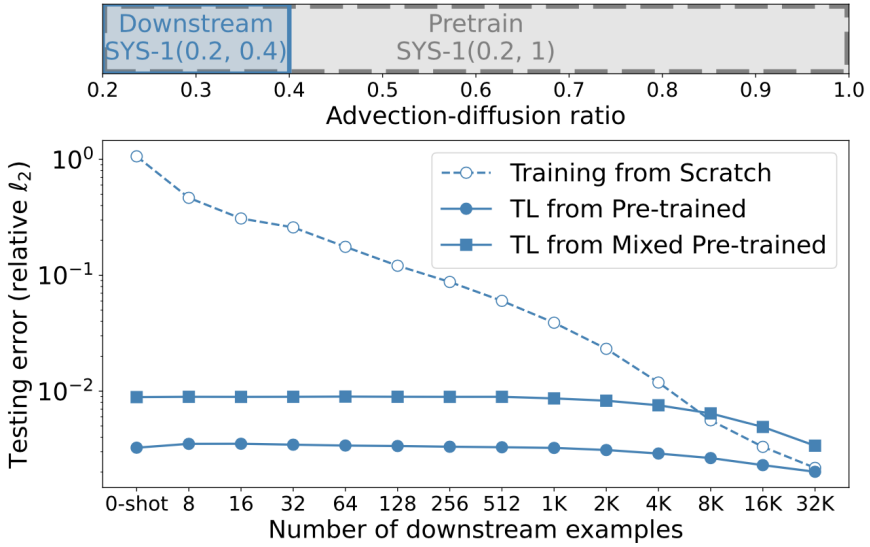
Pre Train On Multiple Systems

$$-\mathrm{div} K \nabla u = f \quad \text{in } \Omega$$



(a) *SYS-1(1,2.5): pre-training using SYS-1(1,5) and mixed dataset*

$$-\mathrm{div} K \nabla u + v \nabla u = f$$



(c) *SYS-2(0.2,0.4): pre-training using SYS-2(0.2,1) and mixed dataset*

Open Problems/Limitations

There are many more open problems/Limitations:

- **Experiments with Real Data:**
  - Our data comes from numerical simulations of dynamical systems with known coefficients. Need to test:
    - Pretraining with real dataset coming from observations (adds lots of interesting moving parts related to sensor noise, uncertainty, ill-posed problems)
    - There is not enough real data set -> Real + Simulated dataset? Need to address questions around domain shift between simulation and senor data
- **NN Architecture:**
  - We did not change the FNO model architecture but is that the right model for all kinds of SciML problems?
    - Need to investigate how the architecture should be changed as the underlying dynamics change
      - Elliptical vs Hyperbolic vs Parabolic PDEs may need different architectures

Outline

- Introduction
- Foundation Models for SciML?
- **Incorporating Physical Laws into Learning**
- Conclusions and Future Work

Incorporate Physical Laws into Learning

- Neural Networks **require a lot of data** to train
- Collecting large scale data is not always possible, for many applications, especially in medical and scientific domain
- However, an important source of data are the **Physical Laws** that govern our world which have been largely ignored in exchange for observed data
- Physical Laws include:
  - Conservation of Mass, Momentum, Energy, etc.

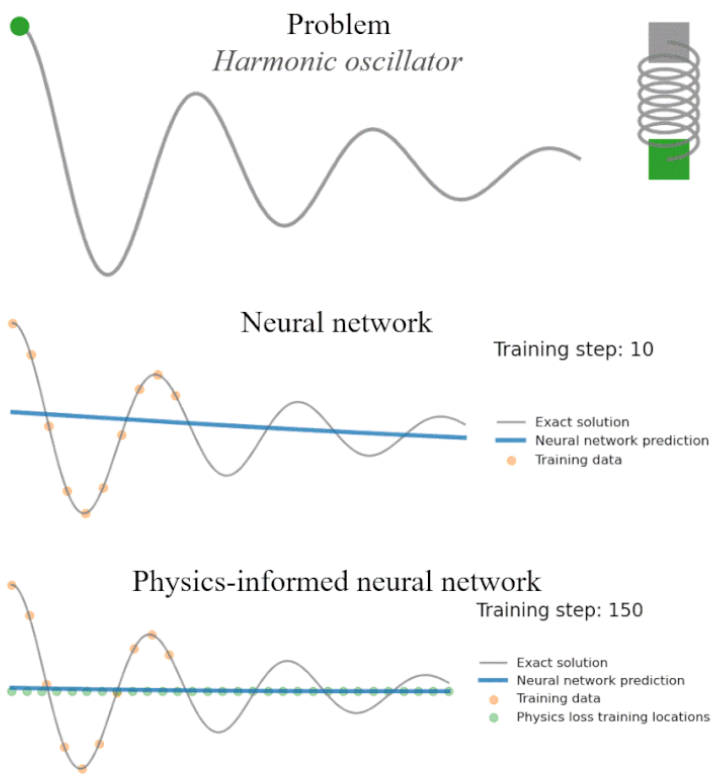
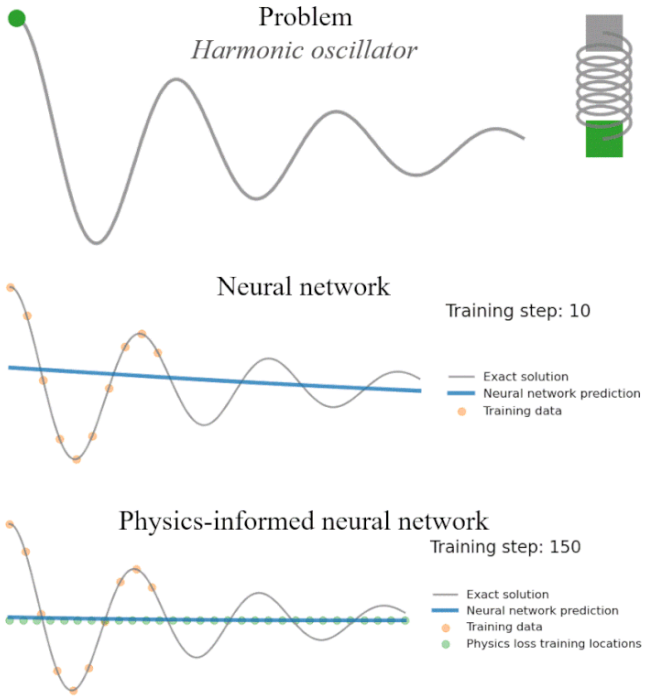


Illustration from Ben Moseley

Incorporate Physical Laws into Learning



$$m\frac{\partial^2 u}{\partial t^2} + \mu\frac{\partial u}{\partial t} + ku = 0$$

$$\min \frac{1}{N}\sum_{i}^{N}(\hat{u} - u_{true})^2 +$$

$$\frac{1}{M}\sum_{j}^{M}(m\frac{\partial^2 u_j}{\partial t^2} + \mu\frac{\partial u_j}{\partial t} + ku_j)^2$$

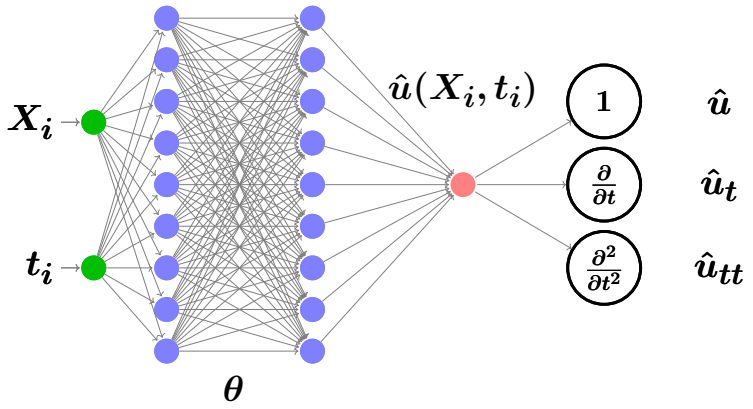


Illustration from Ben Moseley

Physical Laws as Additional Sources of Data

- Other than observed data, we know the invariances that govern physical phenomena
  - Conservation of mass, momentum, energy
  - In many cases, we also have approximate models that can predict the system behavior

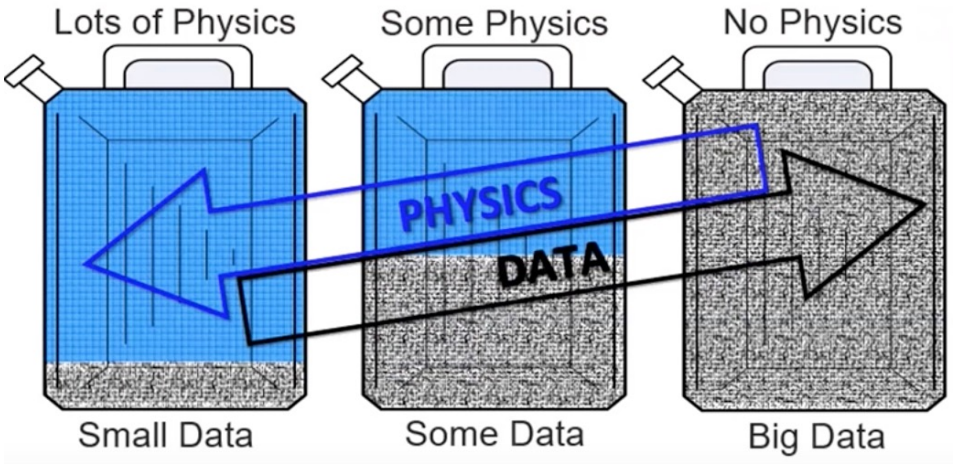
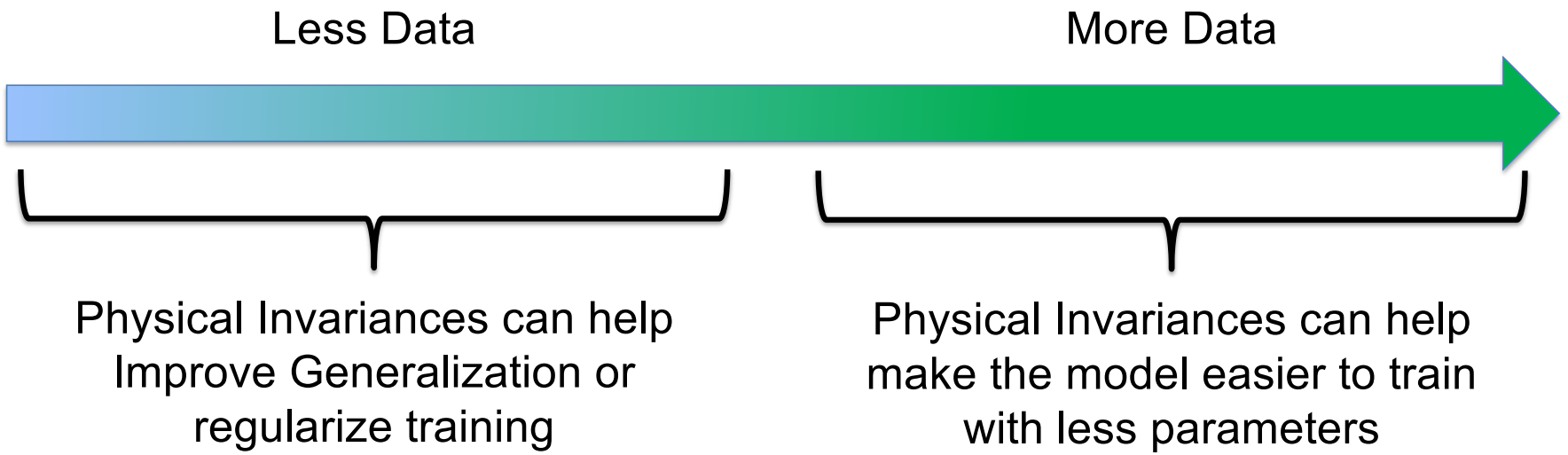


Illustration Credit: Prof. Karniadakis

Physical Laws as Additional Sources of Data

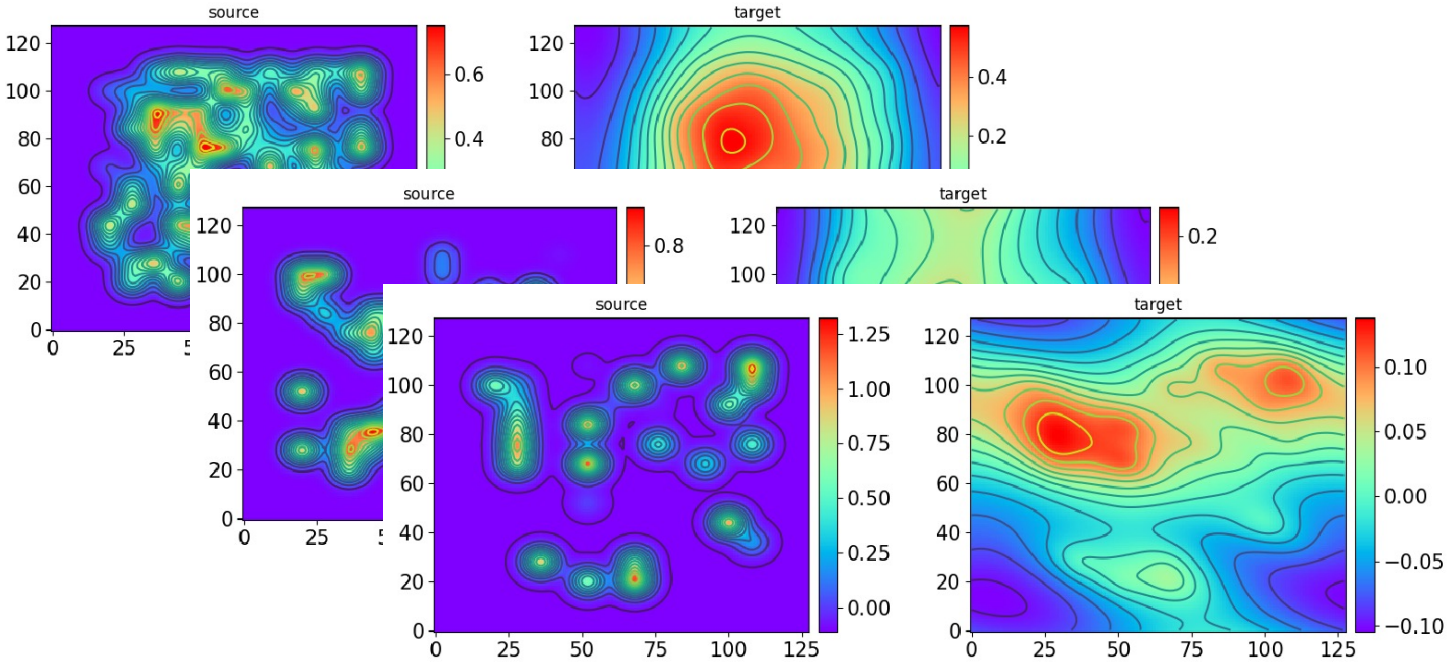
- Other than observed data, we know the invariances that govern physical phenomena
  - Conservation of mass, momentum, energy
  - In many cases, we also have approximate models that can predict the system behavior



The main question is how can we incorporate these invariances into learning?

Methods for Incorporating Physics into Learning

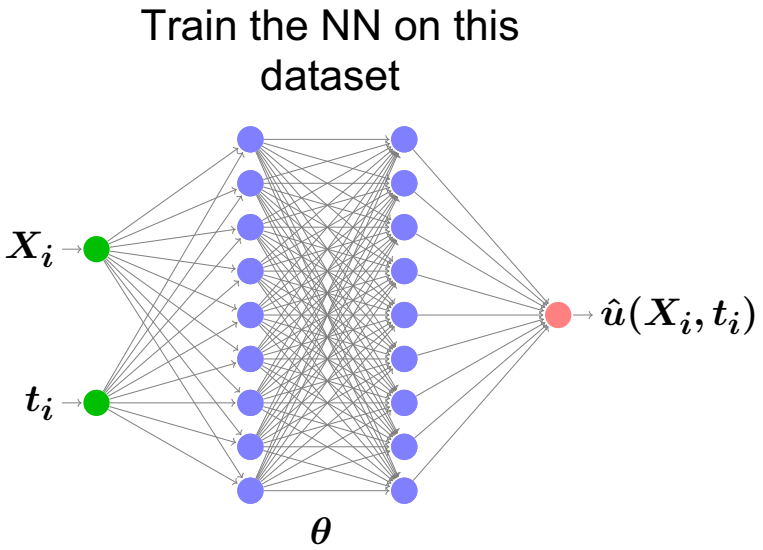
- **Method 1 (Neural Operator): Train on large amount of data and let the NN learn the physics based operators**



[Lu et al., 2019; Bhattacharya et al., 2020; Nelsen & Stuart, 2020; Li et al. 2020, Patel et al., 2021]

Methods for Incorporating Physics into Learning

- **Method 2:** Enforce physical laws as hard constraints either in:
  - NN Architecture: This is an open problem
  - Optimization: Very difficult to train the NN with such constraints



$$\min_{\theta} \mathcal{L} = \sum_{i} \|\hat{u}_i - u_i\|_2,$$

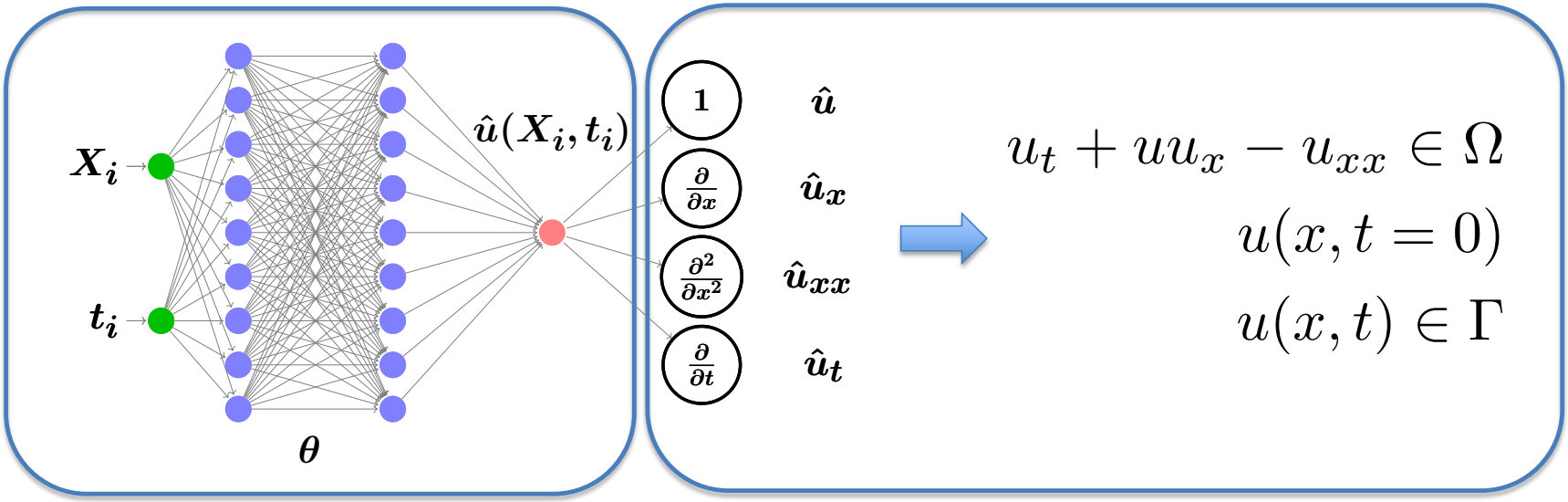
$$s.t. \quad u_t + uu_x - u_{xx} = 0.$$

Xu K, Darve E. Physics constrained learning for data-driven inverse modeling from sparse observations. arXiv preprint arXiv:2002.10521. 2020 Feb 24.

Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Journal of Computational Physics. 2019 Feb 1;378:686-707.

Methods for Incorporating Physics into Learning

- **Method 3:** Use penalty methods and add the **physics model** as a residual to the loss.
- Below we consider Burgers' equation which can be modeled with a PDE (but this approach is applicable even if your model is different than a PDE)



Data Loss Function:

$$\mathcal{L}_u = \|\hat{u} - u\|_2^2$$

Physics Loss Function:

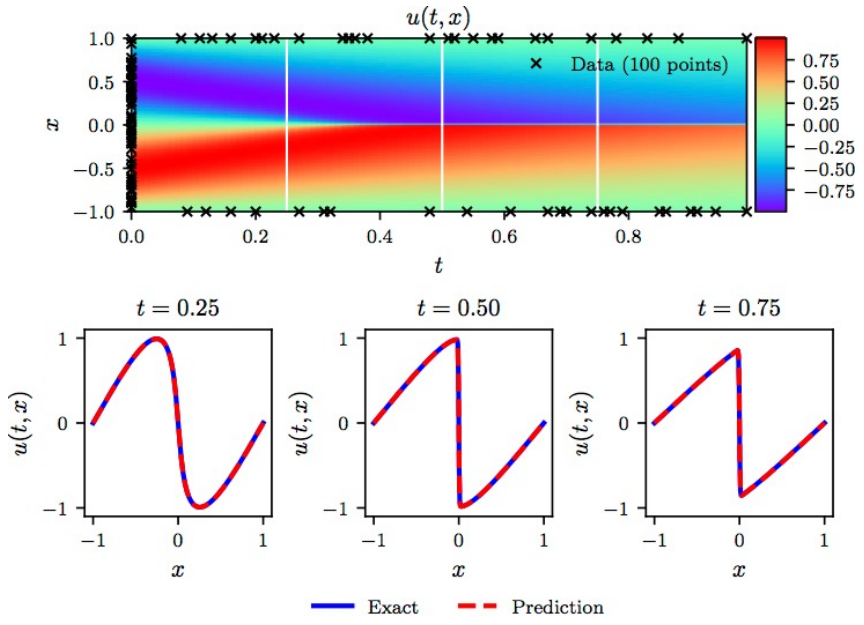
$$\mathcal{L}_\mathcal{F} = \|\hat{u}_t + \hat{u}\hat{u}_x - \hat{u}_{xx}\|_2^2$$

$$\min_\theta \mathcal{L} = \mathcal{L}_u + \lambda_\mathcal{F} \mathcal{L}_\mathcal{F}$$

Physics Informed Neural Networks

- **Method 3:** Use penalty methods and add the PDE residual to the loss.
  - Very easy to implement, and works with any NN architecture
  - Does not require a mesh or a numerical solver for the PDE
  - Can (in theory) work for high dimensional problems, and complex PDEs
    - For example, PDEs containing integral operators which are difficult to solve with finite difference methods.

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$



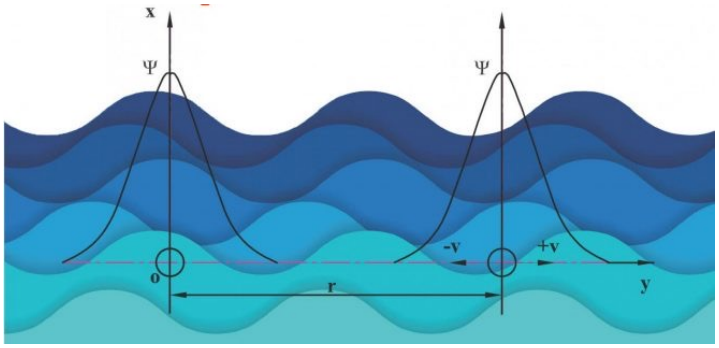
*[Weinan et al. 2017; **Raissi et al. 2019**; Rackauckas et al. 2020; Hennigh et al. 2021; Lu et al. 2021]*

But this is not the entire story

- There are a lot of subtleties in adding a soft-constraint and PINNs actually do not work as well, even for simple problems.
- To study this, we chose three families of PDEs:
  - Advection (aka wave equation)
  - Reaction
  - Reaction-Diffusion

For all of these cases we observed that PINNs fail to learn the relevant physics, since there are many moving parts in this problem

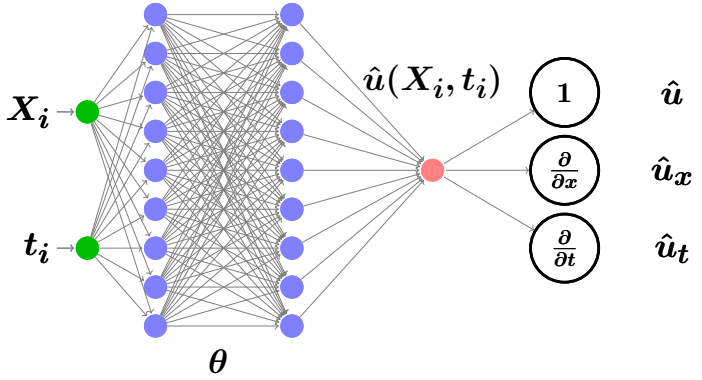
Advection Equation



$$\frac{\partial u}{\partial t} + \beta \frac{\partial u}{\partial x} = 0, \quad x \in \Omega,\ t \in [0, T],$$

Initial condition: $u(x, 0) = sin(x),$

Periodic boundary conditions: $u(0, t) = u(2\pi, t)$



$$\min_{\theta} \mathcal{L} = \lambda_{\mathcal{F}} \|\hat{u}_t + \beta \hat{u}_x\|_2^2 \qquad \text{PDE Residual}$$

$$+ \|\hat{u}(x, 0) - sin(x)\|_2^2 \qquad \text{Initial Condition}$$

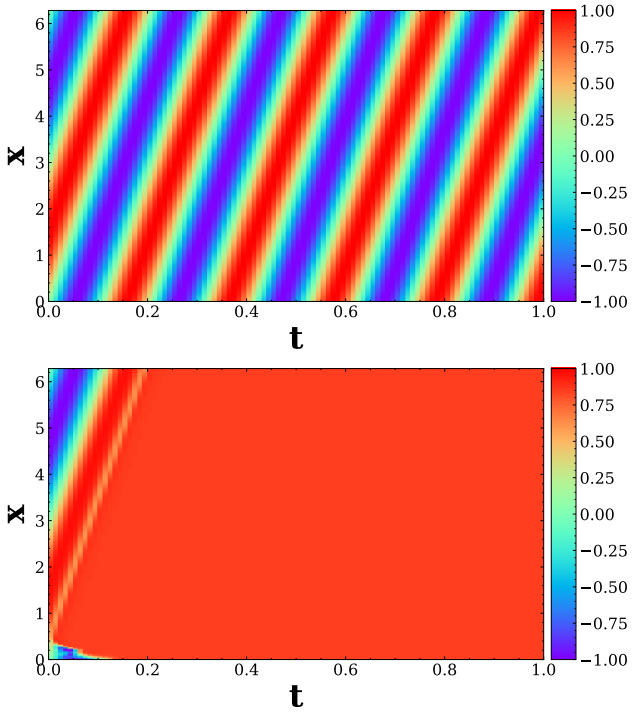
$$+ \|\hat{u}(x = 2\pi) - \hat{u}(x = 0)\|_2^2 \qquad \text{Boundary Condition}$$

Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

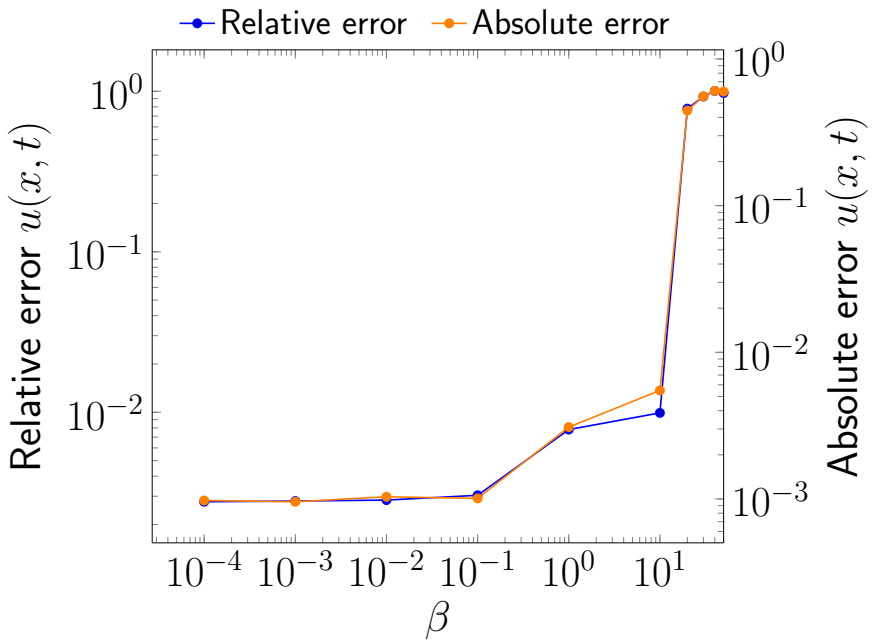
PINN can fail to learn Advection

Exact Solution

PINN Prediction

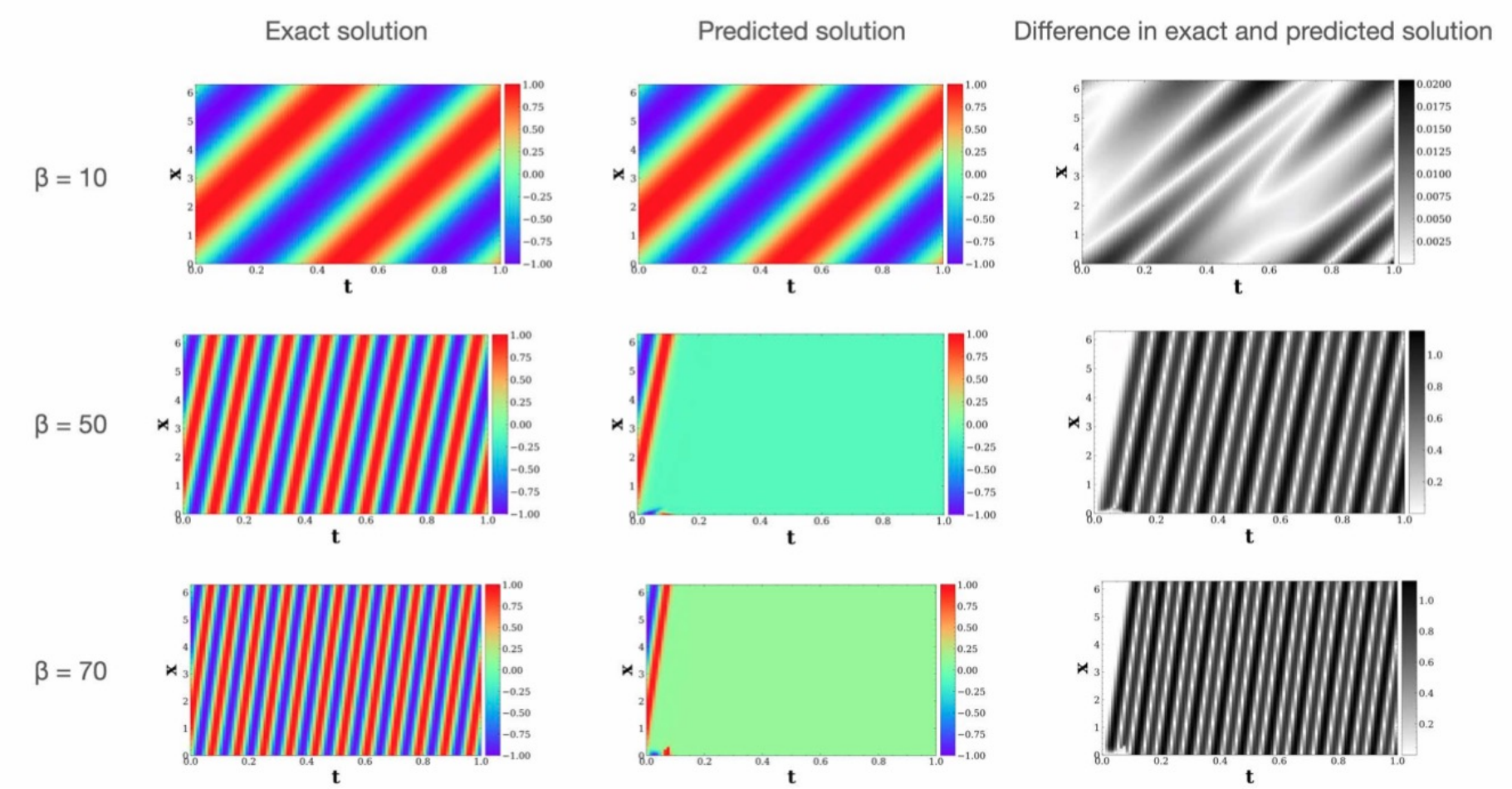


$\beta = 30$



Krishnapriyan AS, **Gholami* A**, Zhe S, Kirby RM, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.*

PINN can fail to learn Advection



Training: Optimization Challenges with PINNs

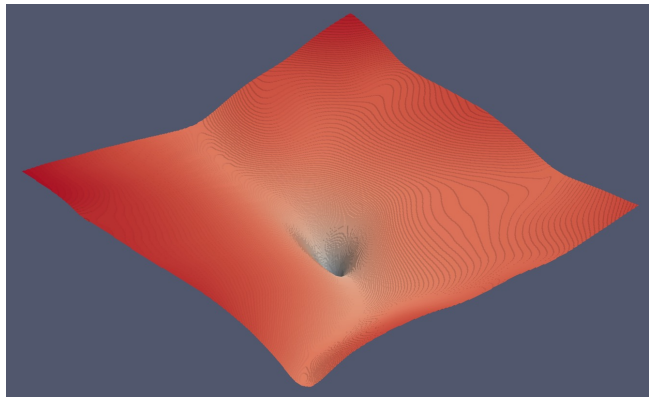
Data Loss Function:

$$\mathcal{L}_u = \|\hat{u} - u\|_2^2$$

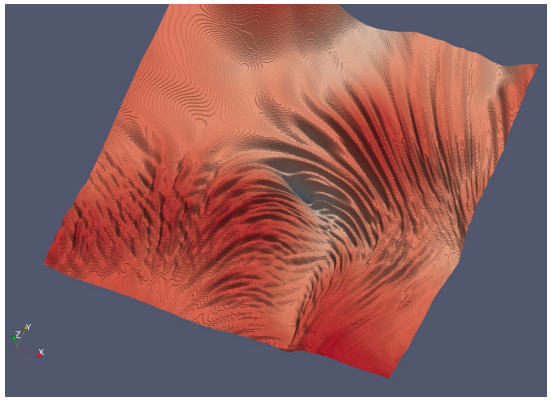
Physics Loss Function:

$$\mathcal{L}_{\mathcal{F}} = \|\hat{u}_t + \hat{u}\hat{u}_x - \hat{u}_{xx}\|_2^2$$

$$\min_{\theta} \mathcal{L} = \mathcal{L}_u + \lambda_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}$$



Without Physics Loss



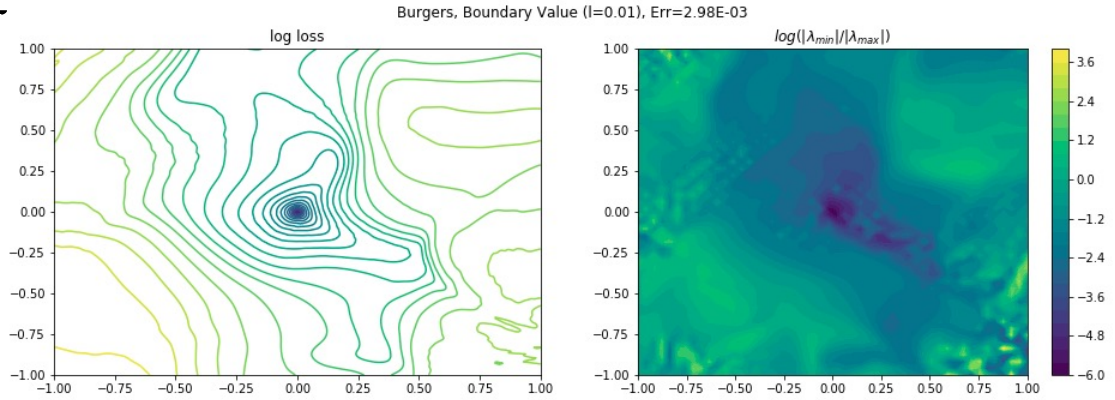
With Physics Loss

Illustration credit: Roman Amici, Mike Kirby

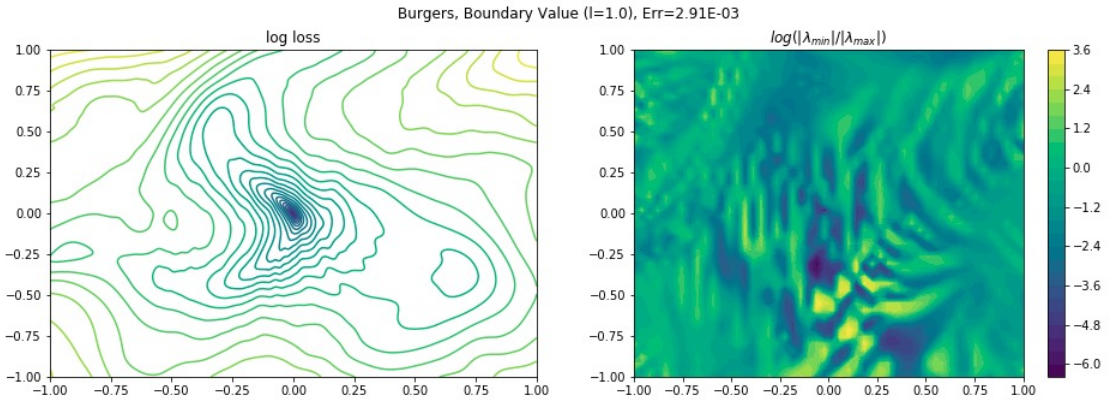
Training: Optimization Challenges with PINNs

$$\min_{\theta} \mathcal{L} = \mathcal{L}_u + \lambda_{\mathcal{F}} \mathcal{L}_{\mathcal{F}}$$

Loss function with 0.01 multiplier for PDE loss



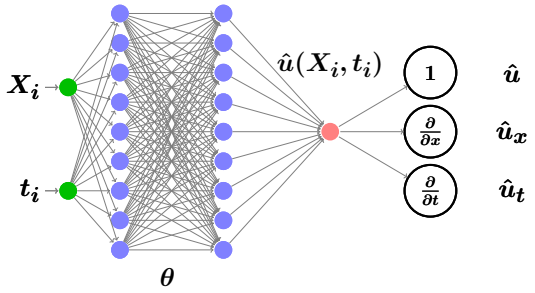
Loss function with 1.0 multiplier for PDE loss



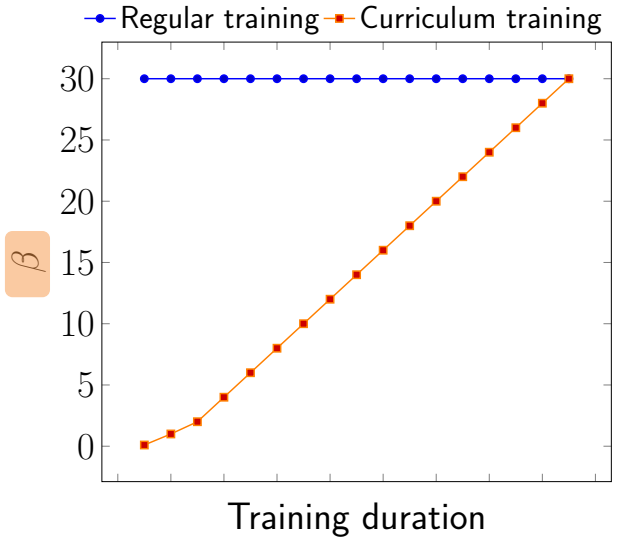
Rethinking PINNs: Curriculum Learning

- The main idea is to start the training with **simple physical constraints** and introduce the complexities iteratively throughout learning
- First let the NN learn the simple problems, before penalizing it for learning the exact PDE

Example: For the advection equation, we start to train the NN with very small velocities, and slowly increase the velocity to the target one



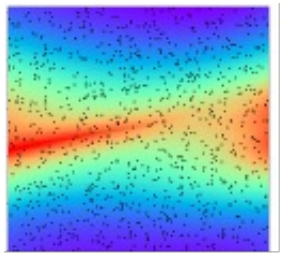
$$\min_{\theta} \mathcal{L} = \lambda_{\mathcal{F}} \|\hat{u}_t + \beta \hat{u}_x\|_2^2$$
$$+ \|\hat{u}(x,0) - sin(x)\|_2^2$$
$$+ \|\hat{u}(x=2\pi) - \hat{u}(x=0)\|_2^2$$

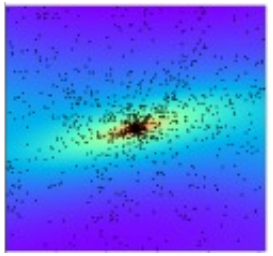


Open Problems

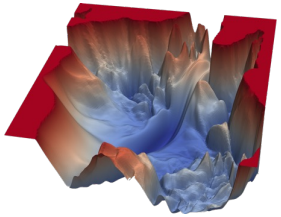
Rethinking the *design*, *training,* and *role of data* for the successful application of neural networks in scientific applications

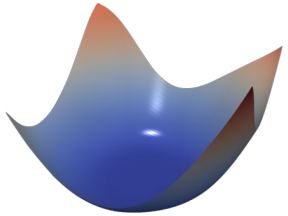
- Adaptive Collocation Points
- PINN Failures: NeurIPS'21
- PyHessian IEEE BigData'20
- Flat/Sharp Minima: NeurIPS'18



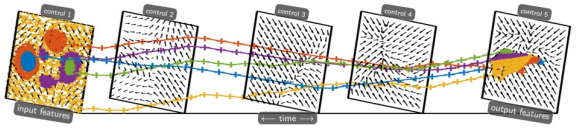


- ANODEV2: NeurIPS'19
- ANODE: IJCAI'19





- On going research on Large Models for Physical Systems



Open Problems/Limitations

- There are many more open problems/limitations:
- **Optimization:**
  - Unlike all other classical ML tasks, PINNs cannot be optimized with mini-batch (SGD, ADAM, etc.) and only works with LBFGS with full batch size
  - This makes training PINNs very slow and hard to optimize

- **NN Architecture:**
  - Same as the previous limitation. The model we considered is the same as the original PINN set up, but different NN architectures may exhibit very different behavior*.
    - *Our preliminary experiments with transformers/attention has not shown improvements wrt optimization challenges

Thanks for Listening

Please reach out if you had any feedback/questions:

amirgh@berkeley.edu

References:

- Subramanian S, Harrington P, Keutzer K, Bhimji W, Morozov D, Mahoney M, Gholami A. Towards Foundation Models for Scientific Machine Learning: Characterizing Scaling and Transfer Behavior. arXiv preprint arXiv:2306.00258. 2023.
- Krishnapriyan A, Gholami A, Zhe S, Kirby R, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. NeurIPS, 2021.