Transfer learning for surrogate models of PDEs

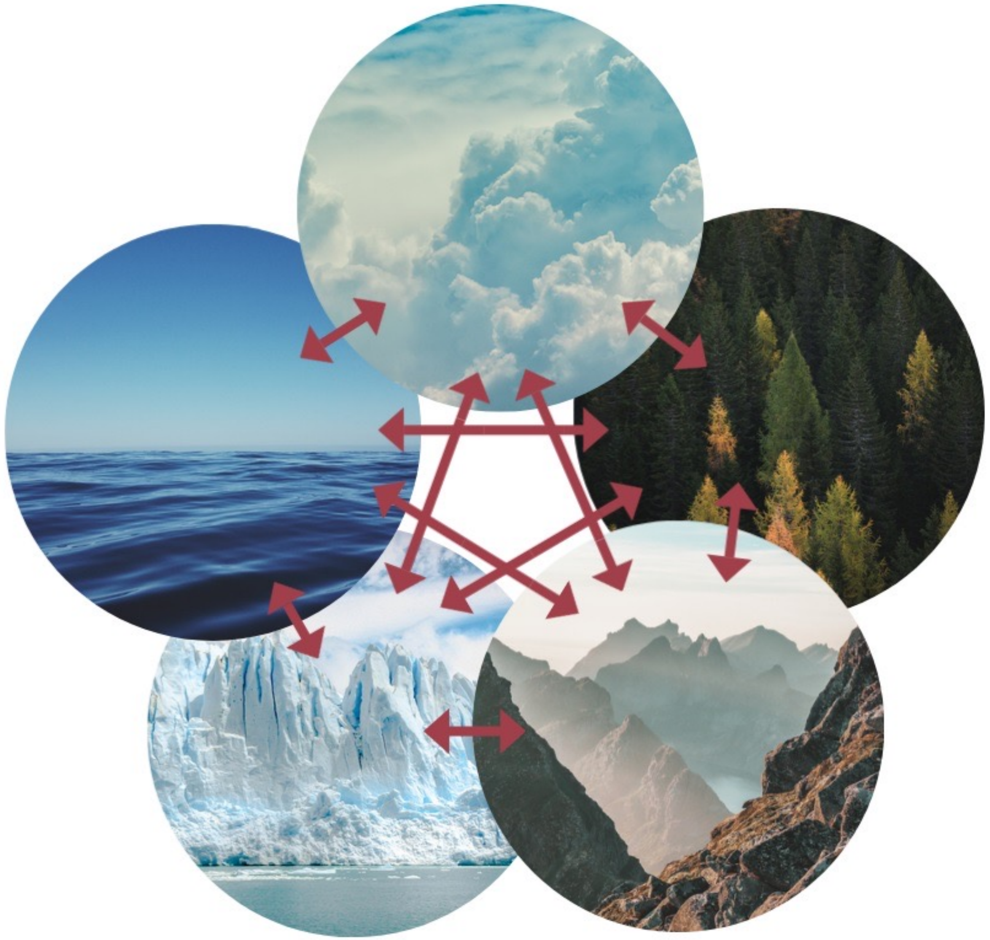
Adrienne Propp

Joint work with Daniel Tartakovsky

Institute of Computational and Mathematical Engineering, Stanford University

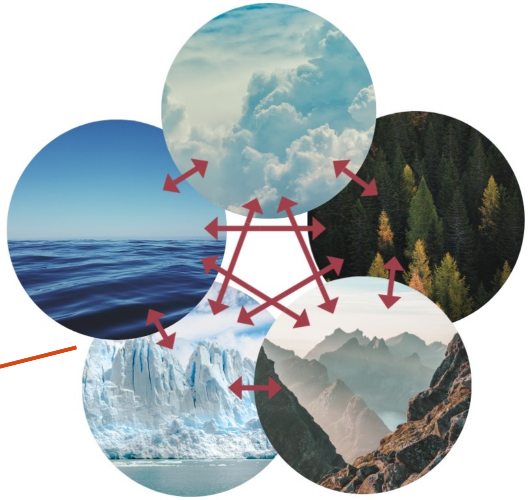
ICME Stanford | Doerr School of Sustainability | Energy Science & Engineering

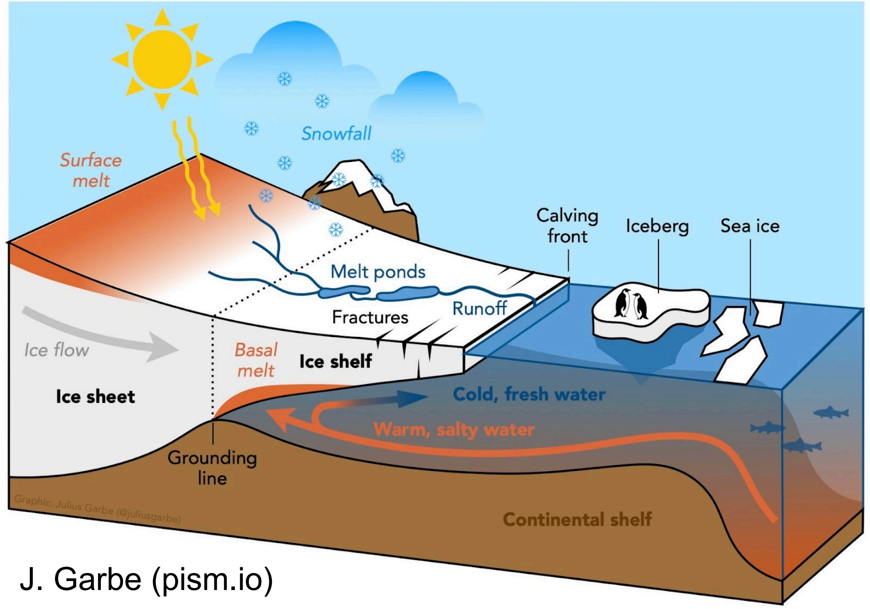
This work is part of a larger effort to develop efficient, scalable techniques for modeling complex, multiscale *systems-of-systems*



SEA-CROGS

This work is part of a larger effort to develop efficient, scalable techniques for modeling complex, multiscale *systems-of-systems*

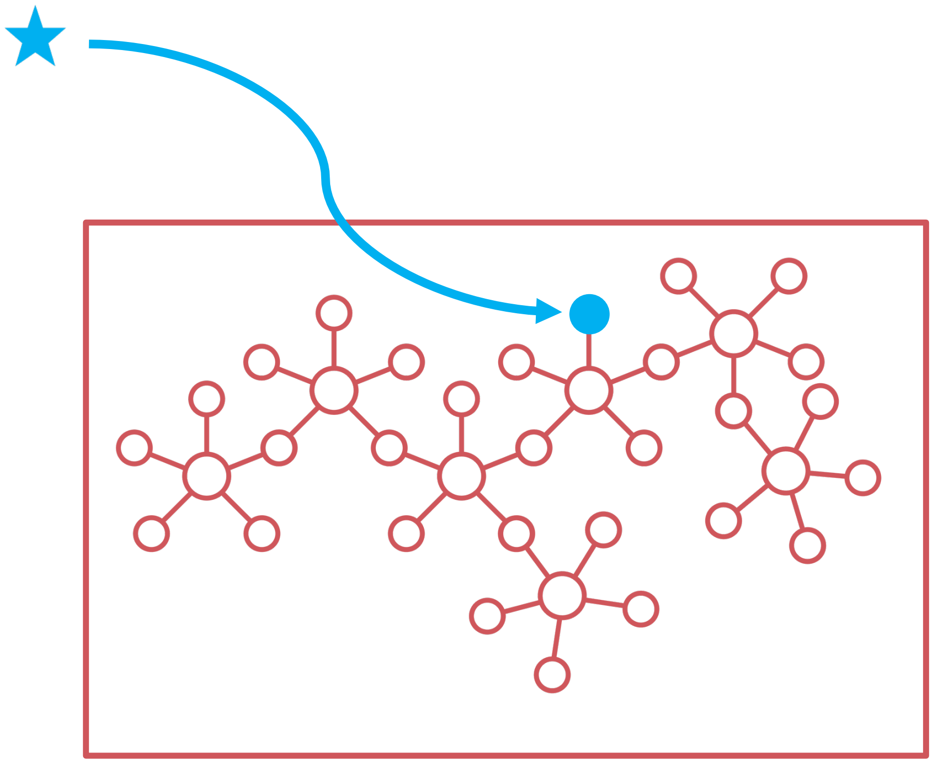


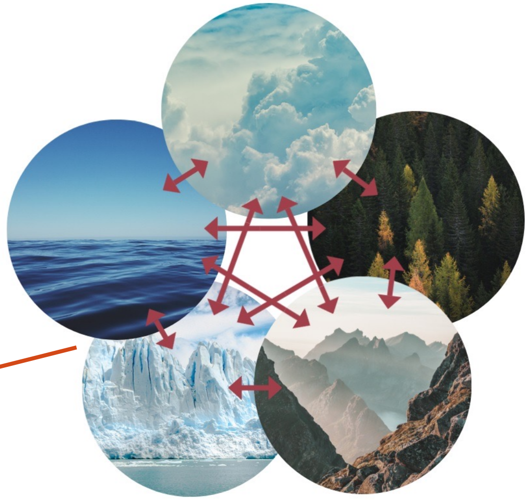


J. Garbe (pism.io)

SEA-CROGS

This work is part of a larger effort to develop efficient, scalable techniques for modeling complex, multiscale *systems-of-systems*





Our goal: develop an efficient input → output mapping for PDE models

SEA-CROGS

CNN-based surrogate models offer a potential solution

- Low-cost approximation of full PDE model
- Captures statistical behavior of system
- Cheaper to run than full model
- Captures input-output relationship

Convolutional Neural Network

- Hidden layers perform convolutions
- Popular for image processing / computer vision tasks
- Good at modeling complex nonlinear processes
- Inexpensive forward pass

Problem:

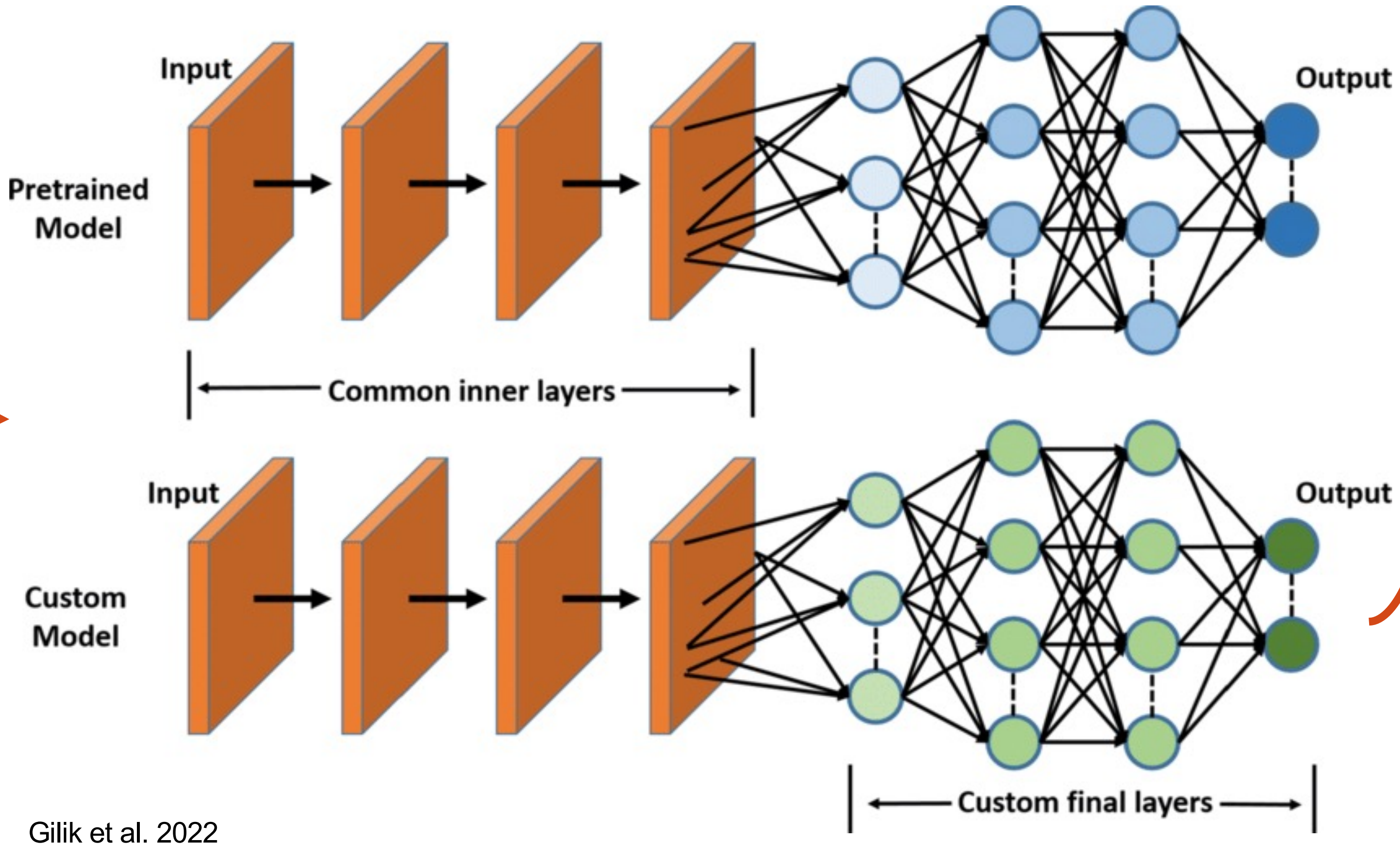
generalizable surrogates must be trained on
a large number of PDE solutions…

…but generating training data can be expensive

We propose **transfer learning on multifidelity data**
as a strategy to reduce the cost of training PDE surrogate models

In **transfer learning**, we apply the knowledge gained from training one model to training another model

Early layers capture coarse features (e.g. edges, shapes) that generalize well to new problems



Gilik et al. 2022

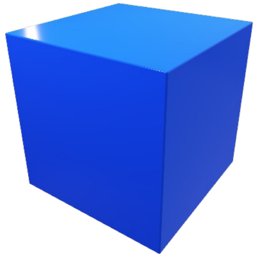
Instead of starting from scratch, initialize with pretrained model, retrain last few layers

→ Reduces training time, resources & data requirements

Multifidelity data spans multiple levels of deviation from the true system of interest

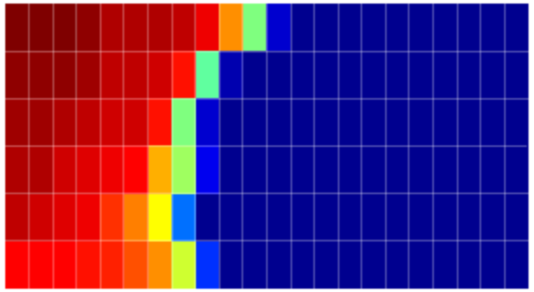
Dimension

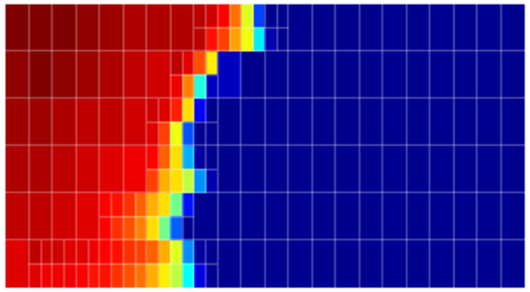
*this work

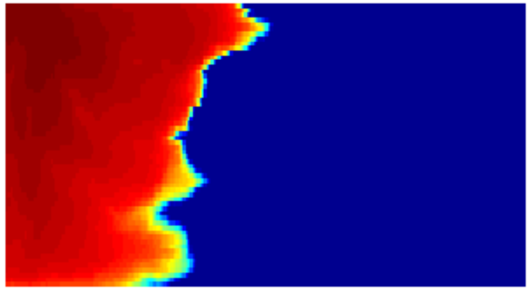


Resolution

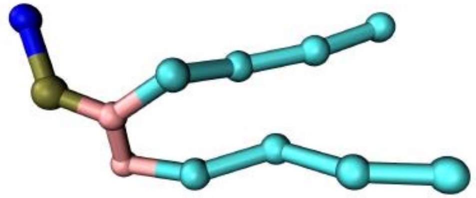
*Song & Tartakovsky, ‘22

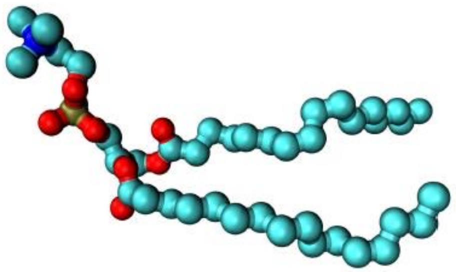


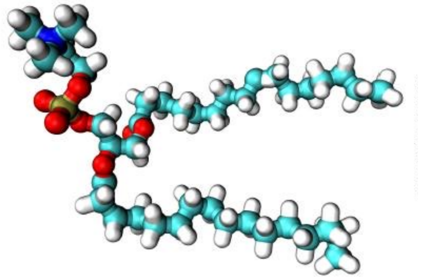




Representation





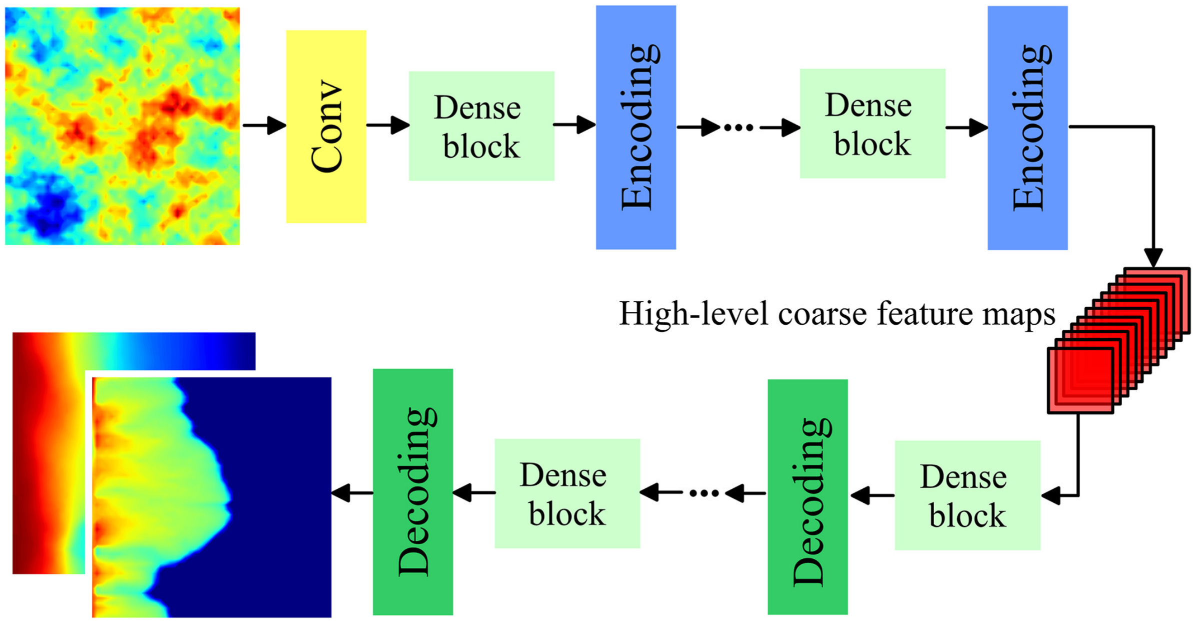


More accurate, provides more information

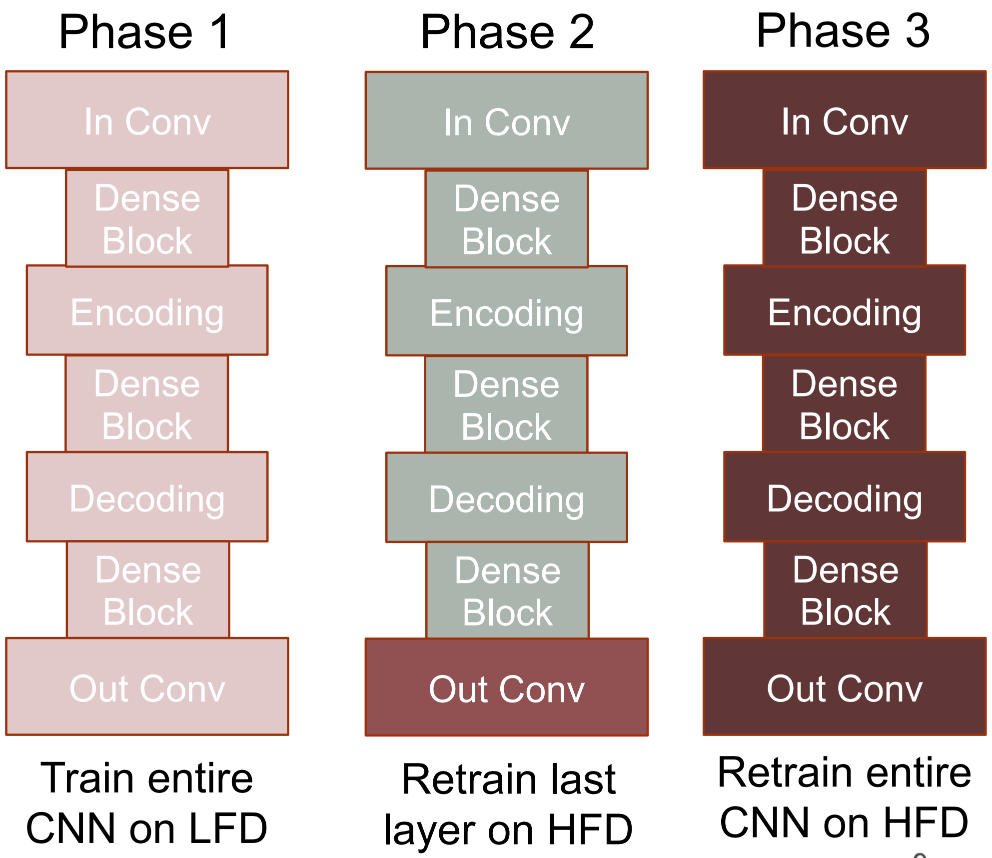
Cheaper to produce

We adopt a dense encoder-decoder network architecture, and train in 3 phases to incorporate multifidelity data

- Same architecture used in Zhu & Zabaras, 2018; Mo et al., 2019; Song & Tartakovsky, 2022
- Dense blocks enhance information propagation, reduce training data requirement



Mo et al., 2019



Our test problem: multiphase flow in a porous medium

- Numerical solutions to multiphase flow problems often expensive due to high degree of nonlinearity, stiffness
- Challenging problem for ensemble-based simulations

$$\phi \frac{\partial S_l}{\partial t} + \nabla \cdot \boldsymbol{v}_l + q_l = 0,$$

$$\boldsymbol{v}_l = -k \frac{k_{rl}}{\mu_l} \nabla P_l$$

$$\boldsymbol{x} \equiv (x_1, x_2)^\intercal \in D, \quad t \in [0, T]$$

$$\frac{\partial P}{\partial x_2} = 0, \quad \boldsymbol{x} \in \Gamma_b \cup \Gamma_t$$

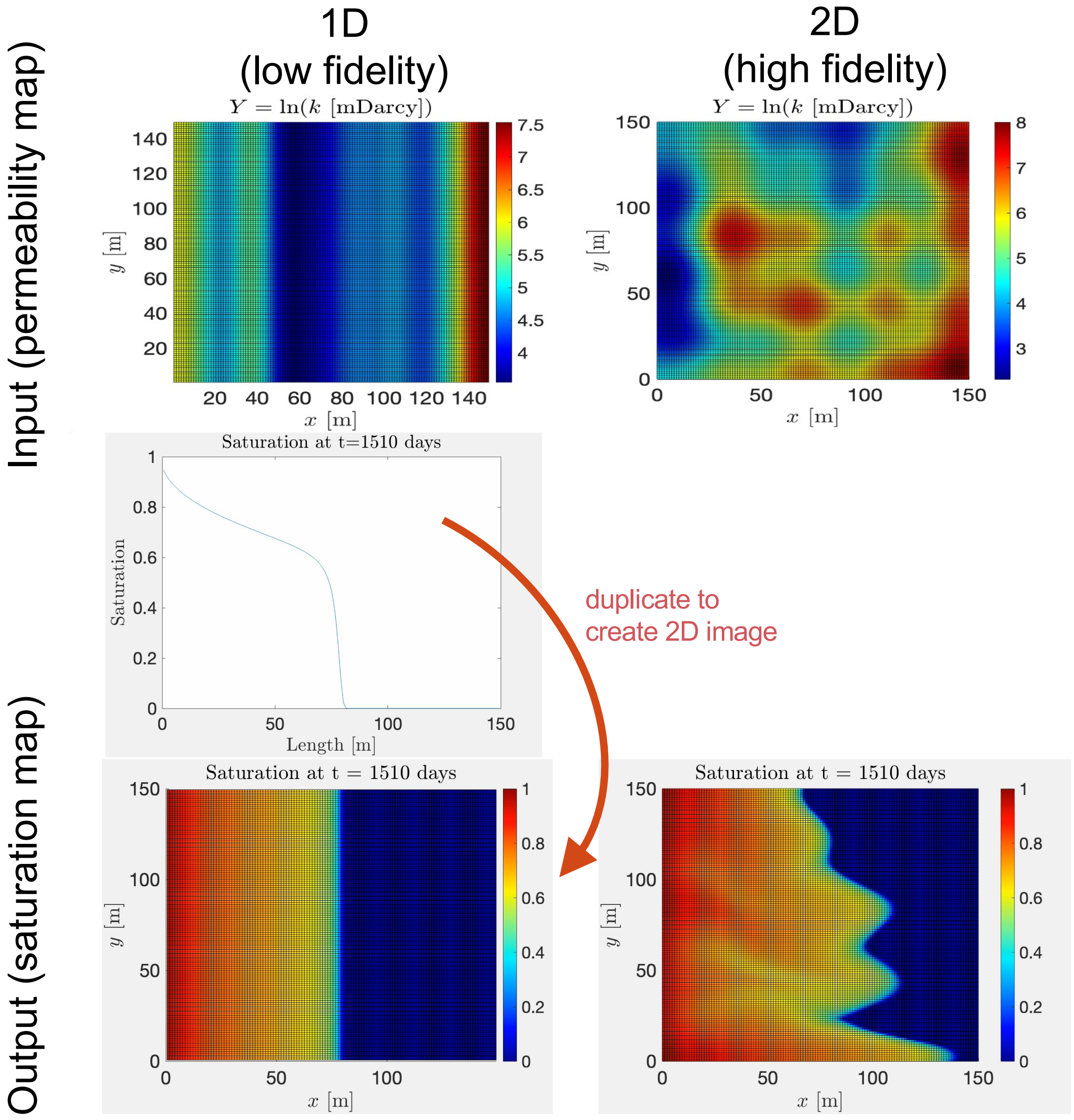
$$P = 10.2, \quad S_1 = 1.0, \quad \boldsymbol{x} \in \Gamma_l$$

$$P = 10.1, \quad S_1 = 0.0, \quad \boldsymbol{x} \in \Gamma_r$$

$$P(\boldsymbol{x}, 0) = 10.1, \; S_1(\boldsymbol{x}, 0) = 0$$

$$P_1 = P_2 \equiv P(\boldsymbol{x}, t), \; S_1 + S_2 = 1$$

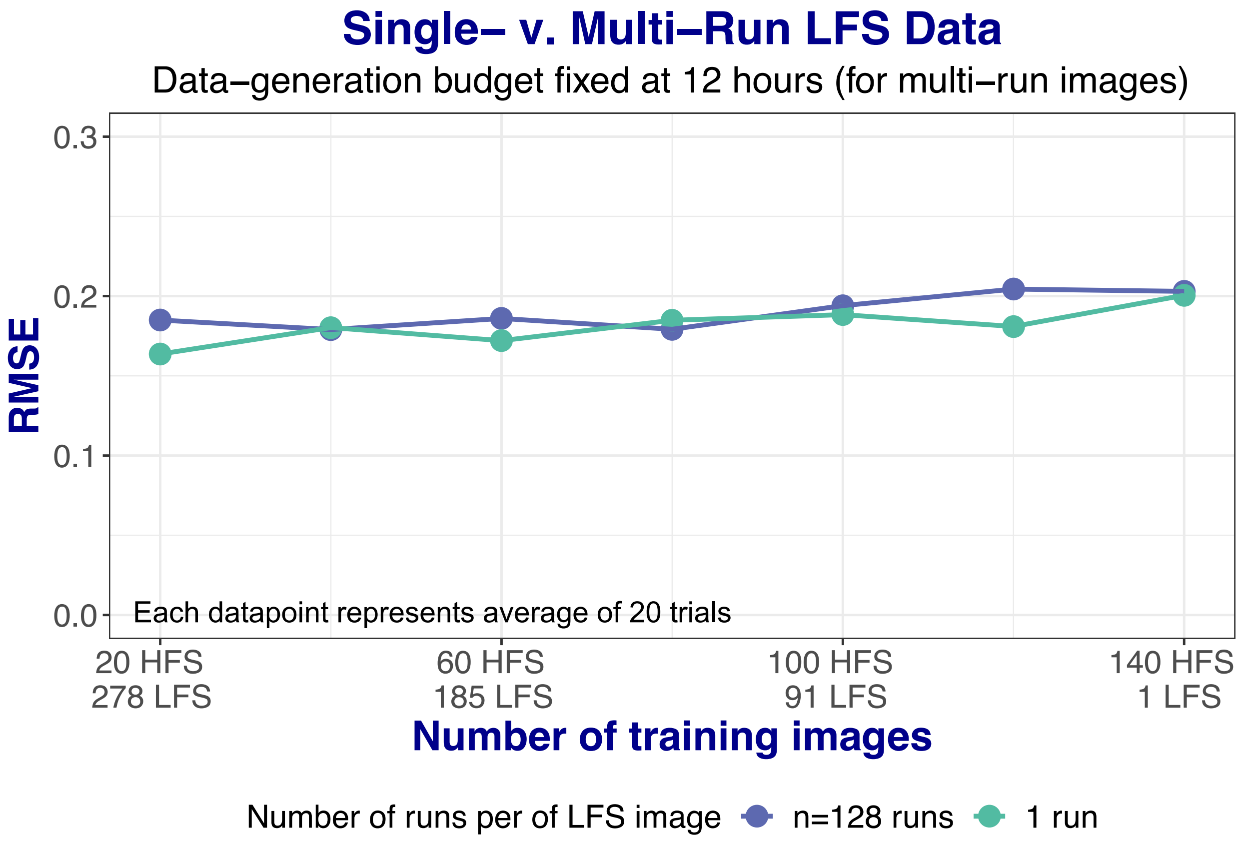
l,	*fluid phase*	$k = k(x)$,	*permeability*
ϕ,	*porosity*	$S_l(x,t)$,	*saturation*
μ,	*viscosity*	$P(x,t)$,	*pressure*
q,	*source/sink*	$v_l(x,t)$,	*velocity*



Observation 1: For a fixed number of training images, model performance was roughly equivalent regardless of whether low-fidelity 1D data contained a single run or n runs

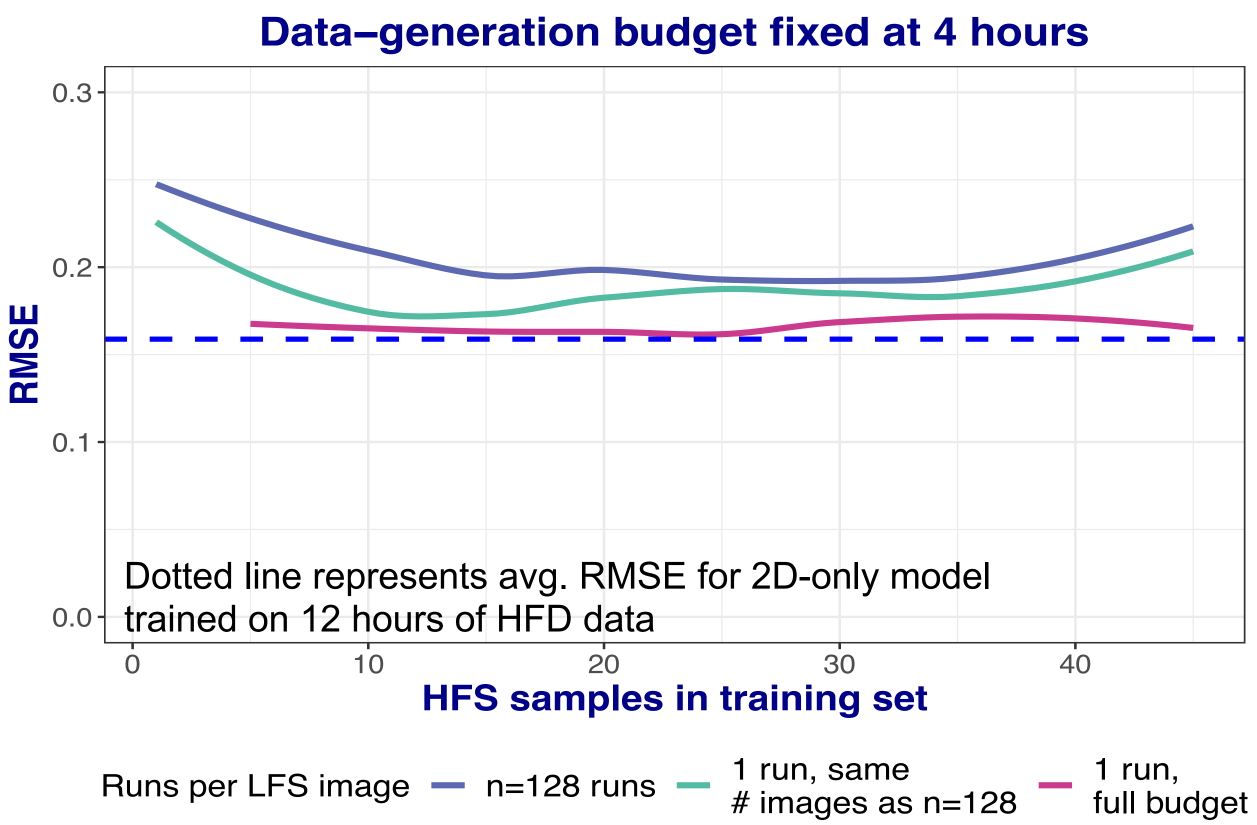
- We **tested two types of low-fidelity (1D) data**:
  - “Single-run”: 1 run per image (e.g. prev slide)
  - “Multi-run”: n=128 runs per image
- Multi-run data:
  - contains 128 times more information,
  - takes 128 times longer to generate,
  - requires 128 times less storage space

  … as single-run data
- For small data generation budgets, both types of 1D data **performed roughly equivalently**
- Somewhere between these extremes may be optimal



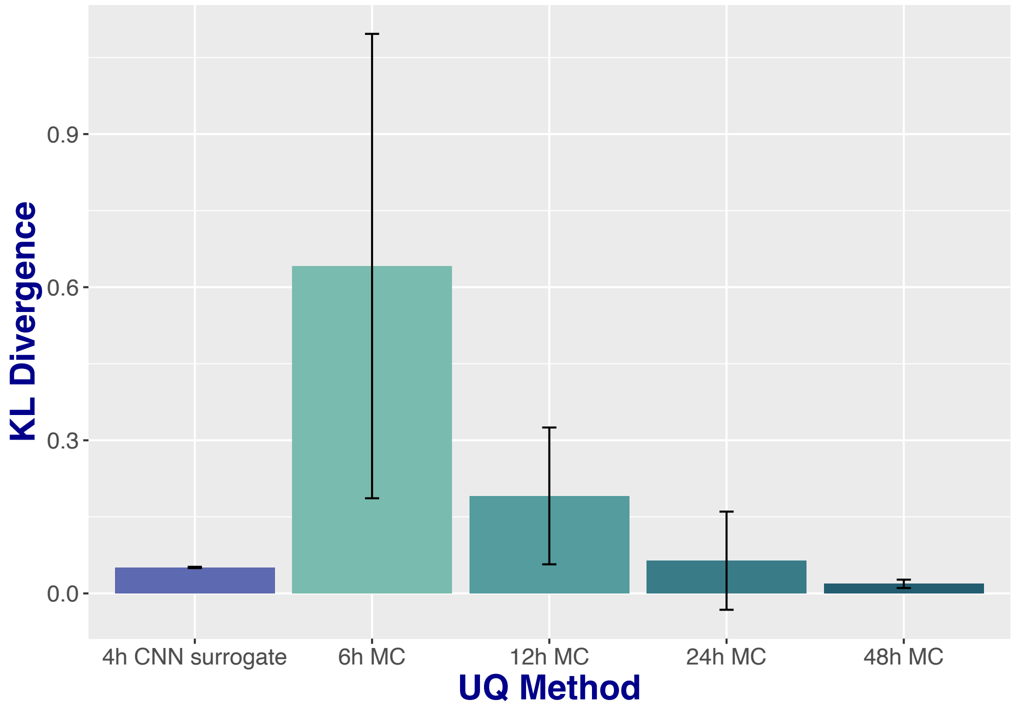
Observation 2: With small data-generation budgets (e.g. 4 hours), models with training sets biased heavily towards either high- or low-fidelity data performed worse than more evenly balanced models

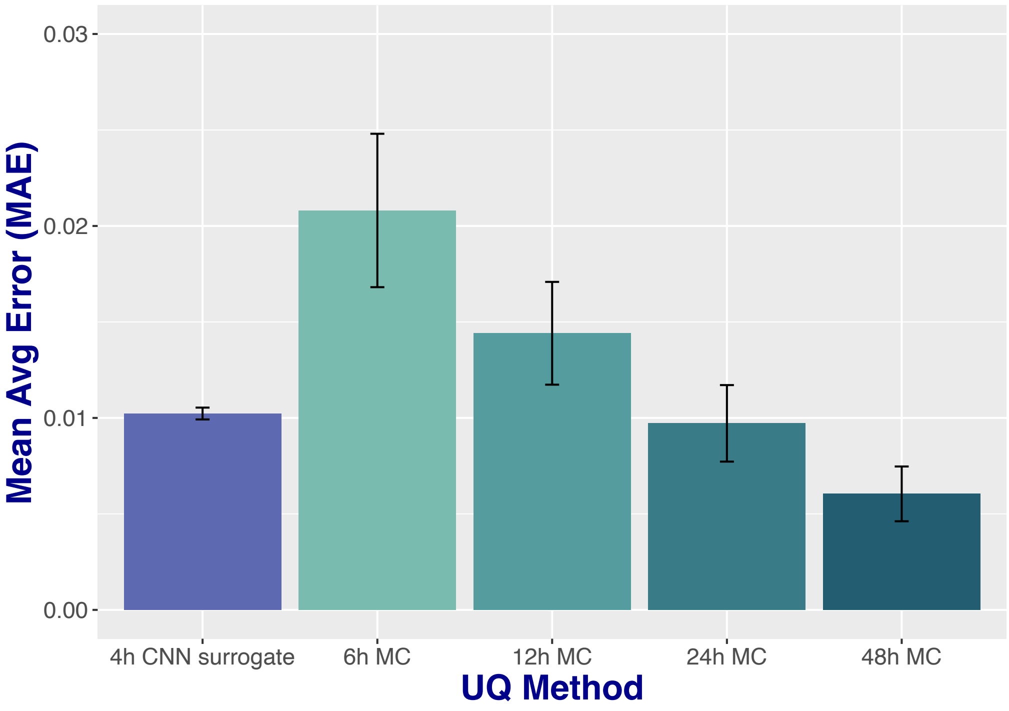
- We tested a data-generation budget of 4 hours with both single- and multi-run LFS data
- We also tested with single-run LFS data, where the number of images matched the budget for multi-run LFS data (green curve)
- MFD models with small data generation budgets performed nearly as well as HFD models with data-generation budgets ~3X larger (dotted line)



Observation 3: In UQ, CNN surrogate trained on 4 hours of MFD performed similarly to Monte Carlo with 24 hours of HFD data, but with lower variance

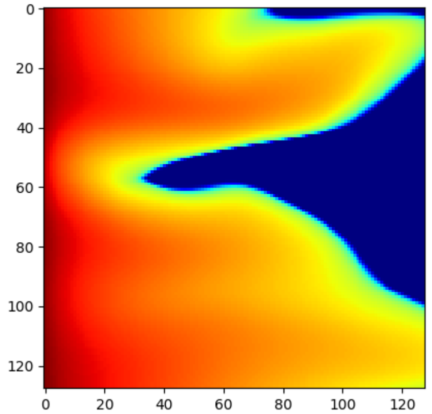
- UQ task: constrain PDF of “breakthrough time” (argmin S_2 *(100, t)* $\geq$ *0.15)*
- CNN surrogate took 19 seconds to complete 2100 forward passes



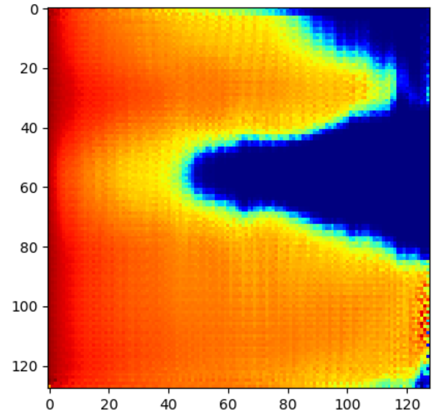


Observation 4: MFD model qualitatively captures flow behavior but ability to capture vertical dynamics is hampered

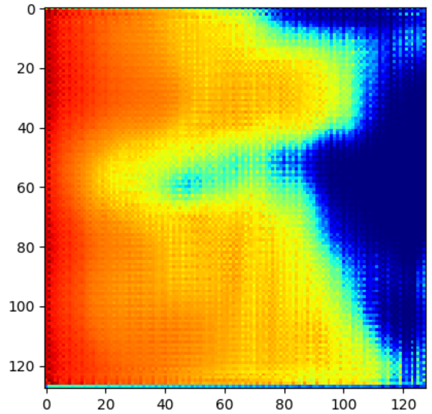
- Future work should test problems with more pronounced dynamics in both directions



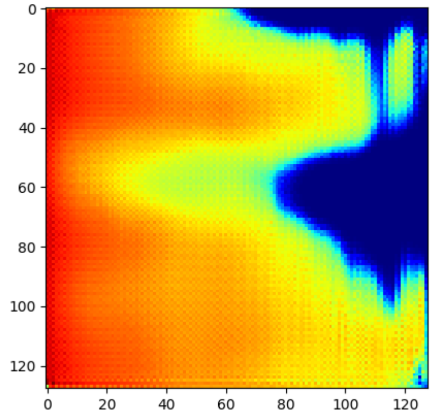
HFD target



4 hours MFD
single-run 1D data
current study CNN surrogate



4 hours MFD
coarse v. fine mesh
Song & Tartakovsky, ‘22



12 hours HFD
2D data only

Summary

- Transfer learning using 1D & 2D MFD was effective for the multiphase flow problem tested
- Low- and high- content 1D data performed similarly, despite >100x difference in runtime & information content
- In UQ tasks, CNN surrogate outperformed MC for up to 6x data generation budget

Next steps

- Investigate whether these results generalizes to strongly multidirectional dynamics
- 2D → 3D
- Extension to graph context
  - e.g. graph pruning, coarse graining
  - Multilevel DDEC (in progress)
  - Integrate with CGC, GINNs