

ML FOR SCIENCE · LECTURE 7

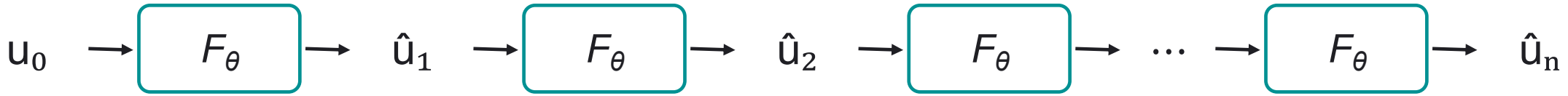
Autoregressive Rollout Models

Aparna Chandramowlishwaran · EECS 195/221 · Fall 2026

Autoregressive model

- A model of one time step, applied to its own output

$$u_{n+1} = F_\theta(u_n), \qquad \hat{u}_n = \underbrace{F_\theta(F_\theta(\cdots F_\theta}_{n \text{ times}}(u_0)))$$

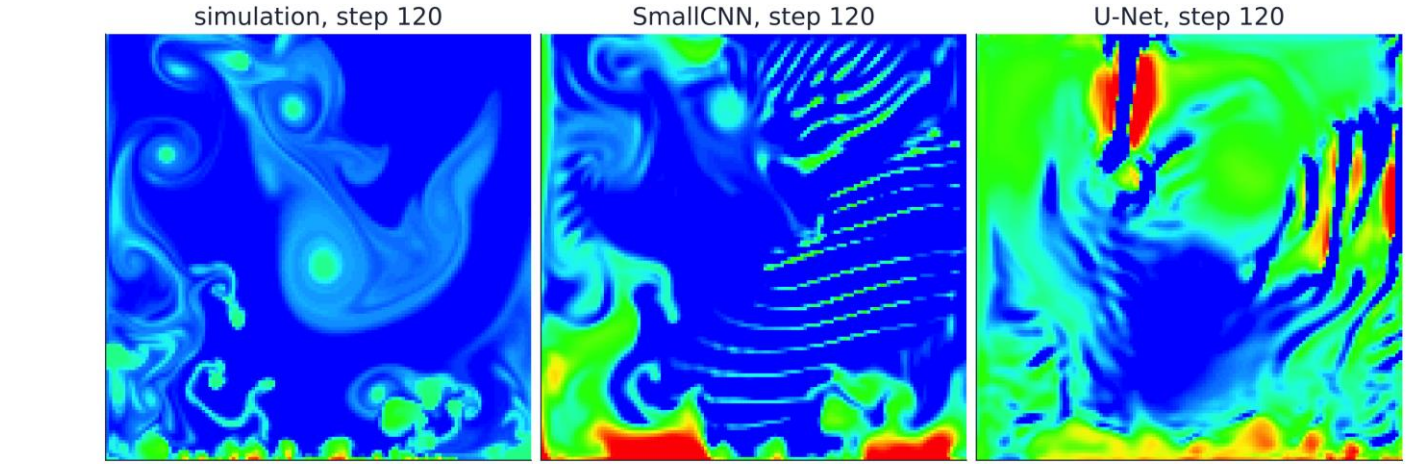


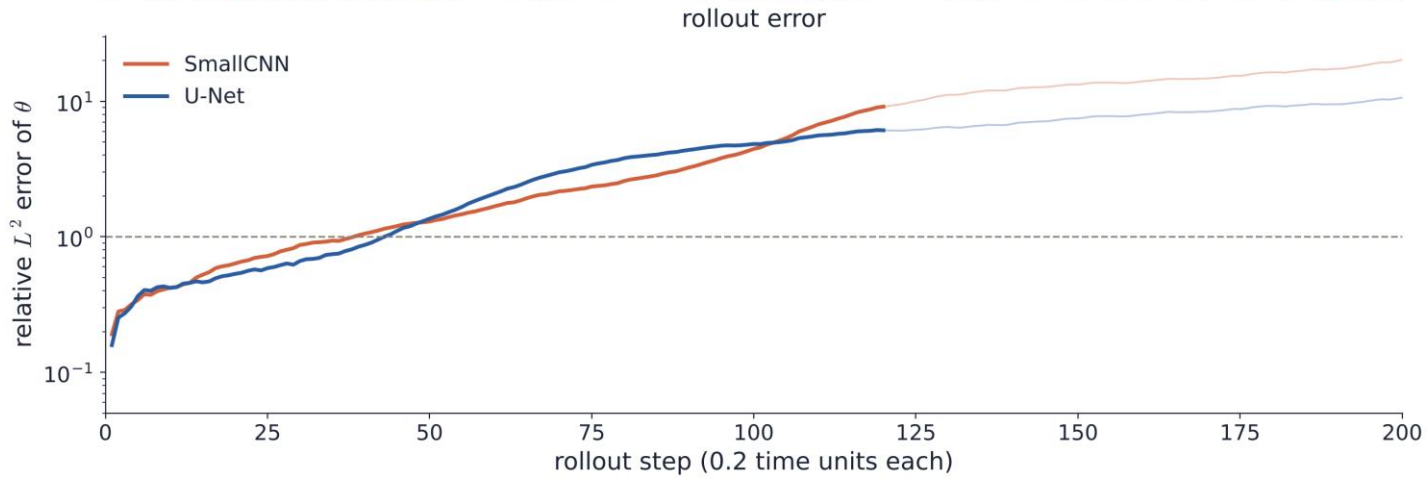
- **One-step error** δ_n: the error of one step forward from the true state. $\delta_n = F_\theta(u_n) - u_{n+1}$
- **Rollout error** e_n: the error after n steps from u_0 $e_n = \hat{u}_n - u_n$
- **Horizon**: how far we roll out (number of steps n or time $n\Delta t$)
- Forward Euler is autoregressive with a fixed update; language models generate text one token at a time

u_n: true state at step n. $\hat{u}_n$: model state after n steps. F_θ: the model, with parameters θ. Δt: time between frames

Rollout of the Lecture 6 models

- Lecture 6: SmallCNN and U-Net, trained on BubbleML to predict the change in temperature over one frame
- Rollout on the test frames: start from the first test frame and feed each predicted output back as input
  - velocity u, v from the simulation at every step, as in Lecture 1
- Both rollouts drift within a few steps and **leave the physical range** (θ outside [0, 1]) after about 50 steps
- The U-Net's lower one-step error gives only a small, temporary advantage





A lower one-step error does not guarantee a better rollout

BubbleML_2, subcooled pool boiling, FC-72, Twall = 106 ° C, 128 x 128, frames 0.2 apart. Trained on the first 600 frames, tested on the last 200.

Growth of rollout error

- Linearize the model around the true state:

$$e_{n+1} = F_\theta(u_n + e_n) - u_{n+1} \;\approx\; J\, e_n + \delta_n, \qquad J = \frac{\partial F_\theta}{\partial u}$$

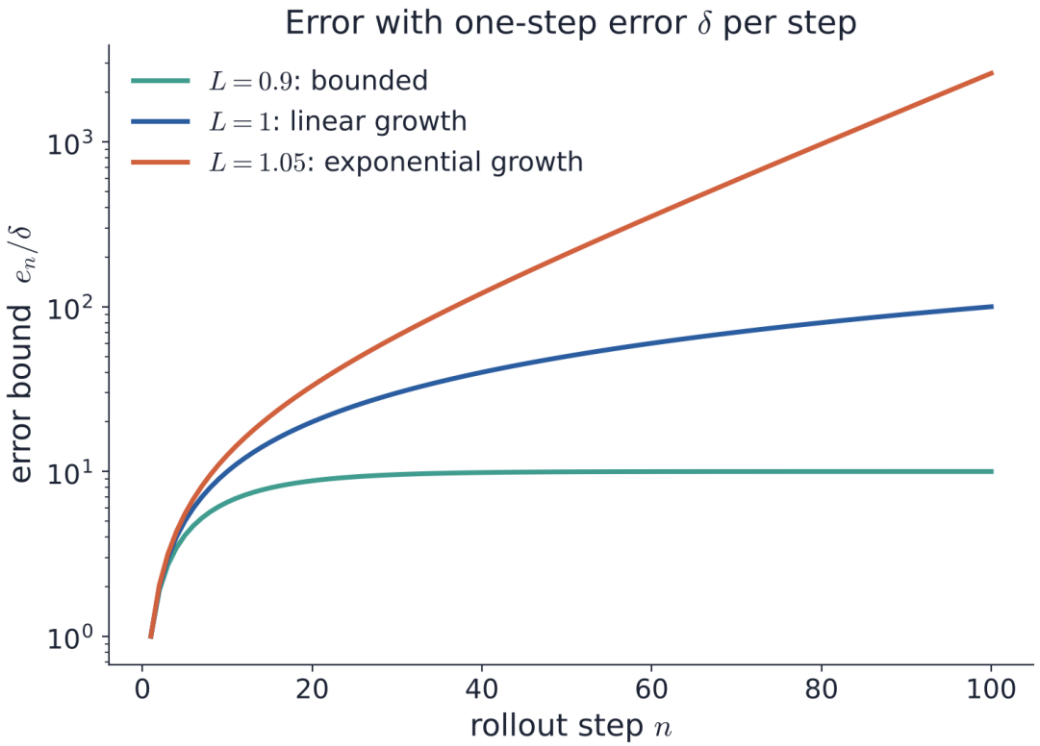
- J: how the model's output changes with its input

$$e_n \approx \sum_{k=0}^{n-1} J^{\,n-1-k}\, \delta_k$$

- Each δ_k is multiplied by J at every later step

$$\|e_n\| \le \delta\,(1 + L + \cdots + L^{n-1}), \qquad \|J\| \le L, \quad \|\delta_k\| \le \delta$$

- L < 1: bounded. L = 1: linear growth. L > 1: exponential

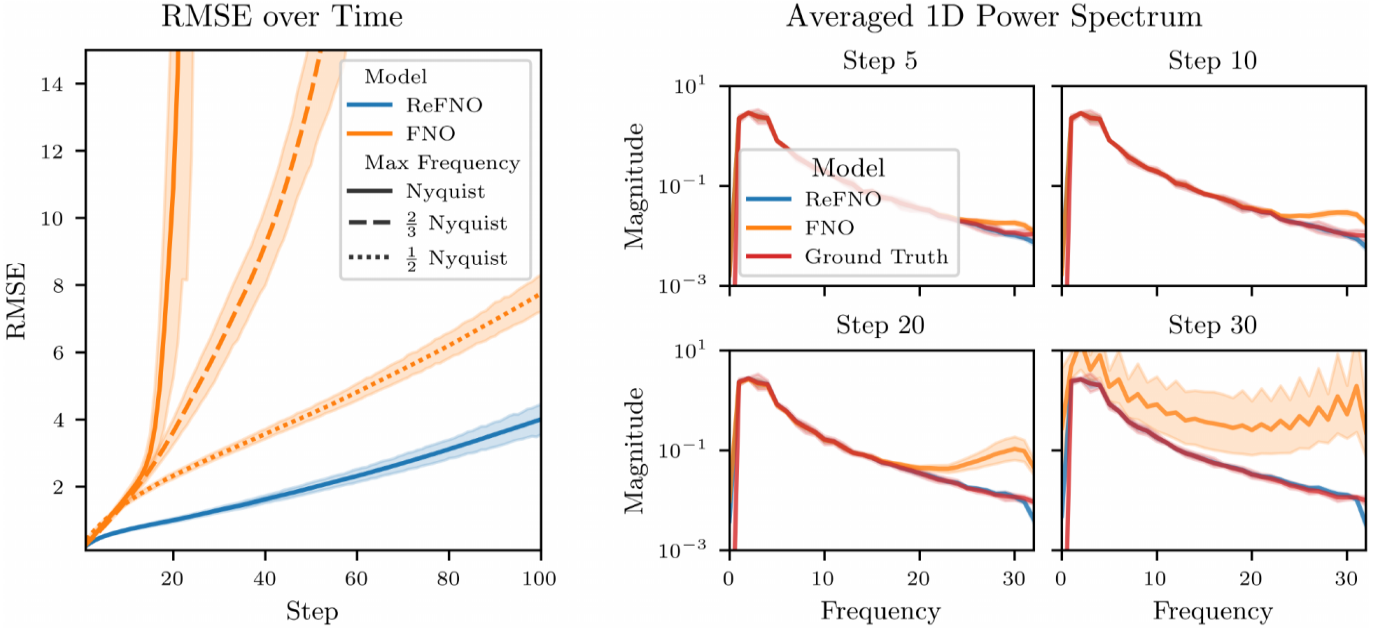


Rollout error depends on the one-step error *and* on how much the model amplifies errors

J: Jacobian of F_θ with respect to its input.

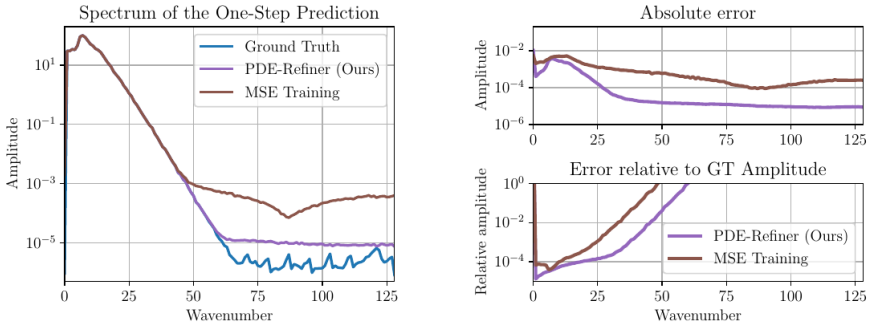
Rollout error in Fourier space

Fourier neural operator on 2D Navier-Stokes (McCabe et al.)



- Left: FNO rollout error grows fastest when it keeps the most modes. Right: its spectrum (**orange**) departs from the truth at high frequency first, and the departure spreads with the step

Kuramoto-Sivashinsky (Lippe et al.)



(b) Frequency spectrum (c) One-step error

- (b) One-step predictions match the truth only at low wavenumbers, where the energy is
- (c) Above that, the error is as large as the field itself (relative error near 1)
- MSE weights each wavenumber by its energy, so low-energy, high-k modes are learned least (spectral bias)

Left: McCabe, Harrington, Subramanian & Brown, Towards Stability of Autoregressive Neural Operators, TMLR 2023, Fig. 1 (ReFNO: their restructured model).
Right: Lippe, Veeling, Perdikaris, Turner & Brandstetter, PDE-Refiner, NeurIPS 2023, Fig. 1(b, c)

Distribution shift in rollout

- **Teacher forcing**: during training, the input is always the true state (Williams & Zipser 1989)
- In rollout, the input is the model's own previous output. It drifts away from the training inputs: **distribution shift**
- The model's errors are larger on inputs it was not trained on, which pushes the state further away
- BubbleML rollout: after 1 to 20 steps, small scales have too little energy; by step 100, too much at every scale
- **Imitation learning**: with one-step error ε, total error over T steps can grow as $T^2\varepsilon$. Training on the model's own states gives $T\varepsilon$ (Ross & Bagnell 2010)

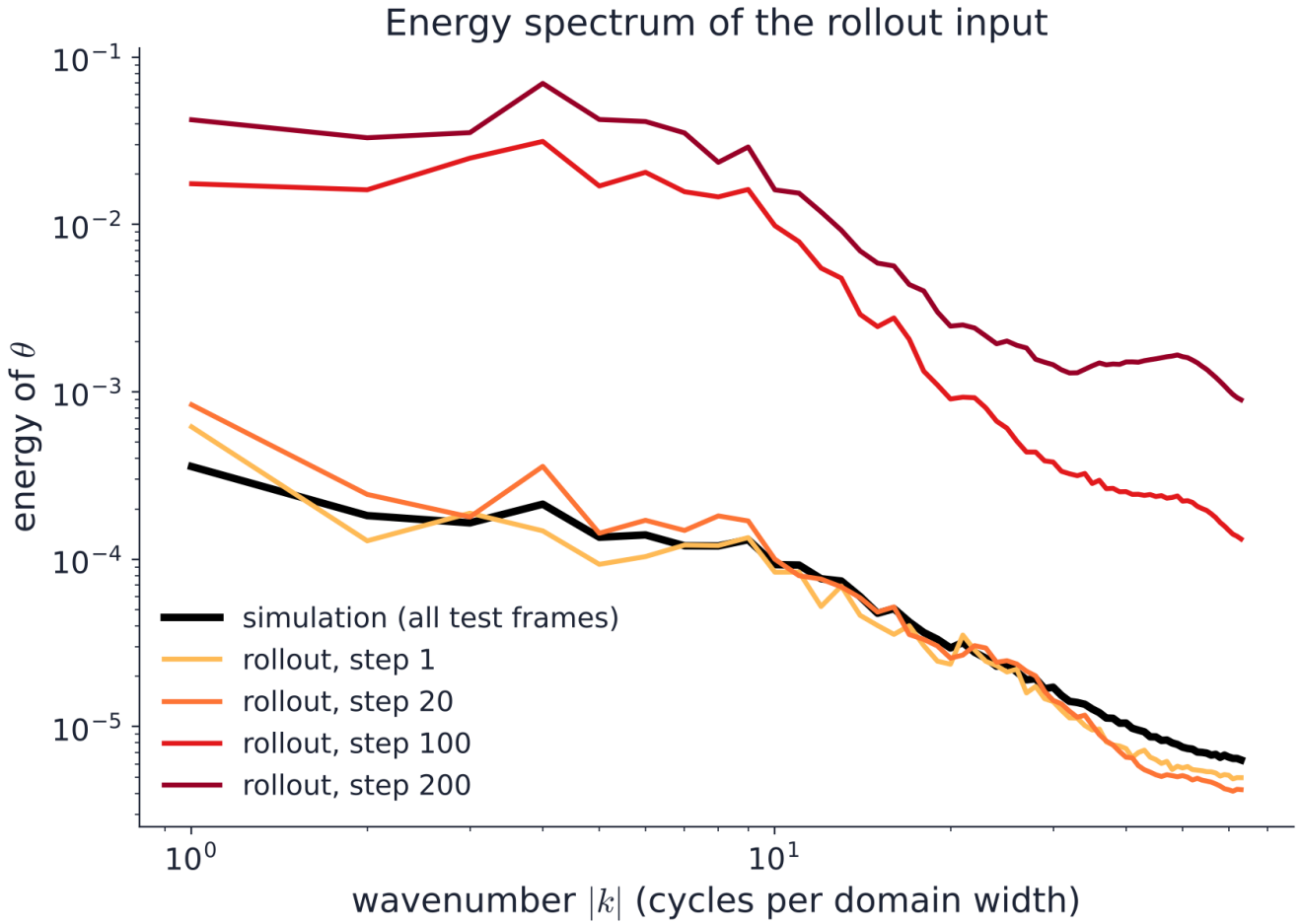


Figure: energy spectrum of θ, the input to the next step, from the rollout of a U-Net that predicts θ, u and v, at several steps; black: simulation (same frames). Ross & Bagnell, AISTATS 2010. Williams & Zipser, Neural Computation 1989

Energy spectra of GraphCast forecasts

- **GraphCast**: 6-hour steps, trained with MSE, fine-tuned on rollouts of up to 12 steps
- **Top**: power spectrum of temperature at two pressure levels. **Bottom**: power relative to the ERA5 targets
- At wavelengths of about 100 to 1000 km, GraphCast has **too little power**, and it drops further with lead time (darker blue)
- The authors attribute this to MSE training: where the future is uncertain, the best MSE forecast is smooth
- BubbleML rollout (previous slide) also lost small-scale energy in its first steps

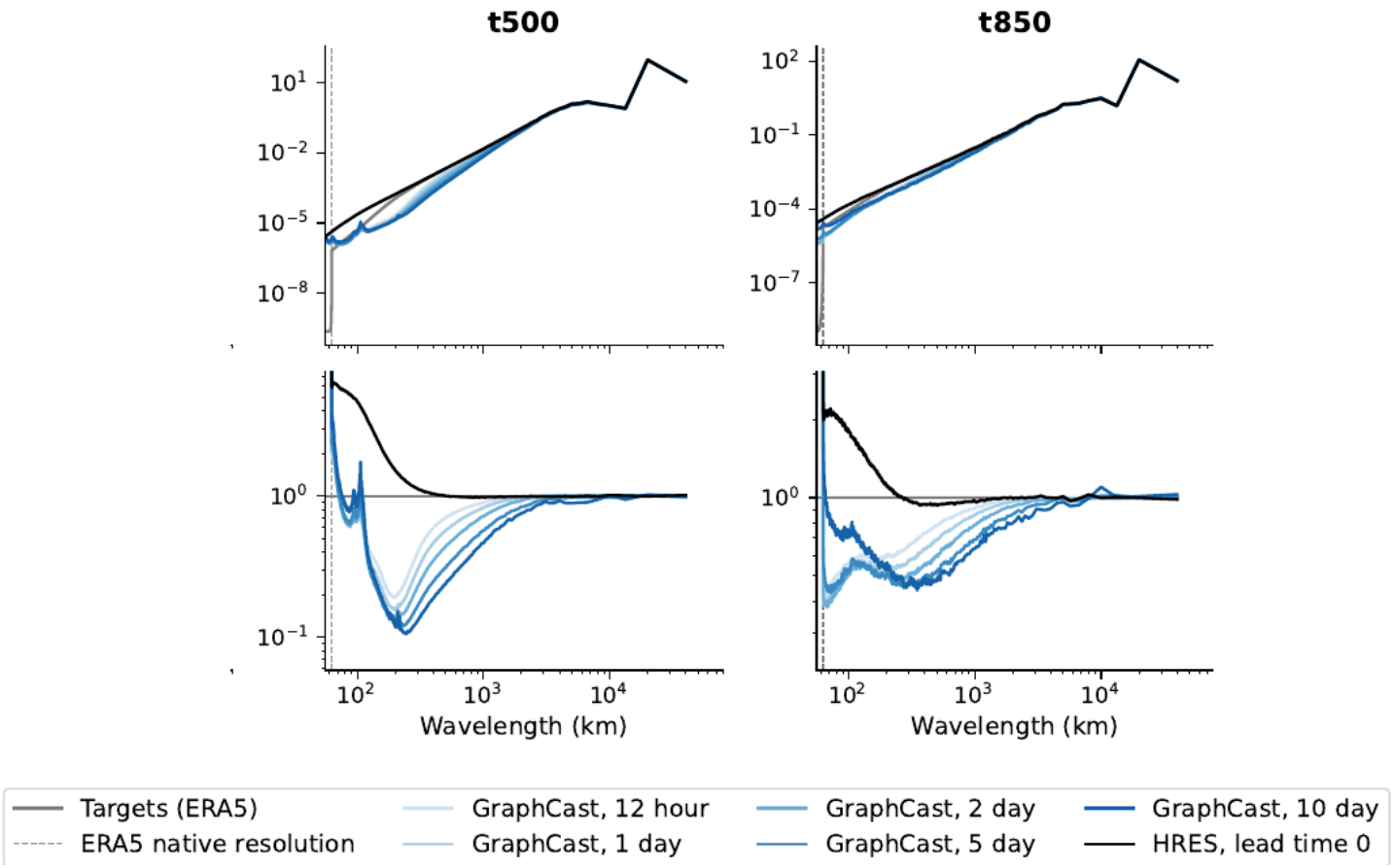


Figure: Lam et al., Learning skillful medium-range global weather forecasting, Science 2023. HRES: ECMWF operational forecast

Noise injection

- Add random noise to the input during training. The target stays the true next state:

$$\mathcal{L}_0 = \left\| F_\theta(u_n + \varepsilon) - u_{n+1} \right\|^2, \qquad \varepsilon \sim \mathcal{N}(0, \sigma^2)$$

- The model learns to map slightly wrong states back toward the data: it learns to correct small errors
- Sanchez-Gonzalez et al. (ICML 2020): random-walk noise on the input velocities of a particle simulator
- Panels g, h: one-step error **grows** with σ; rollout error is lowest at an **intermediate σ**
- **Cost**: the same as one-step training
- **Limitation**: Gaussian noise is not representative of model errors (for example blur)

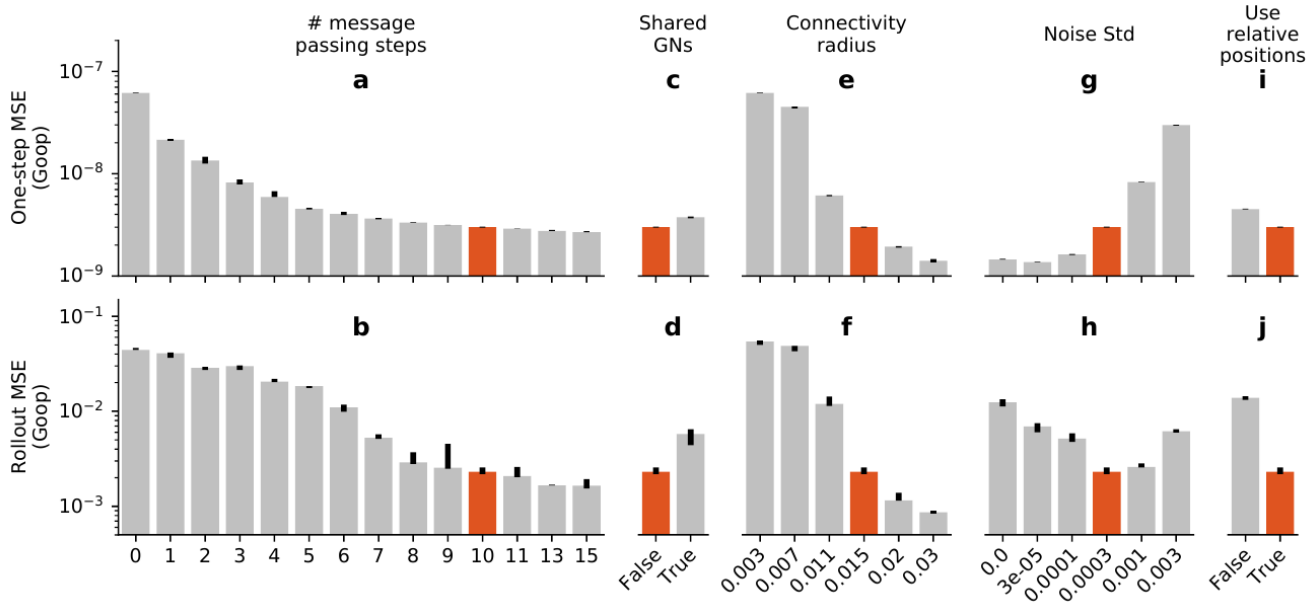


Figure: Sanchez-Gonzalez, Godwin, Pfaff, Ying, Leskovec & Battaglia, Learning to Simulate Complex Physics with Graph Networks, ICML 2020, Top row: one-step MSE; bottom row: rollout MSE (Goop dataset, log scale). Red: their default. Noise: panels g, h

Unrolled training

- Rollout K steps during training and compare every step with the truth:

$$\mathcal{L}_0 = \sum_{j=1}^{K} \left\| F_\theta^j(u_n) - u_{n+j} \right\|^2$$

- **Backpropagate through all K steps**: the model sees its own errors, and the gradient includes how the first step changes the error K steps later
- **Cost**: K forward and backward passes; memory for K steps of activations
- The gradient passes through K Jacobians and can explode or vanish, as in recurrent networks
- Um et al. (NeurIPS 2020): unrolled training with a differentiable solver. Kochkov et al. (PNAS 2021): unrolled training on 2D turbulence. GraphCast: fine-tuned with rollouts of up to 12 steps (3 days)

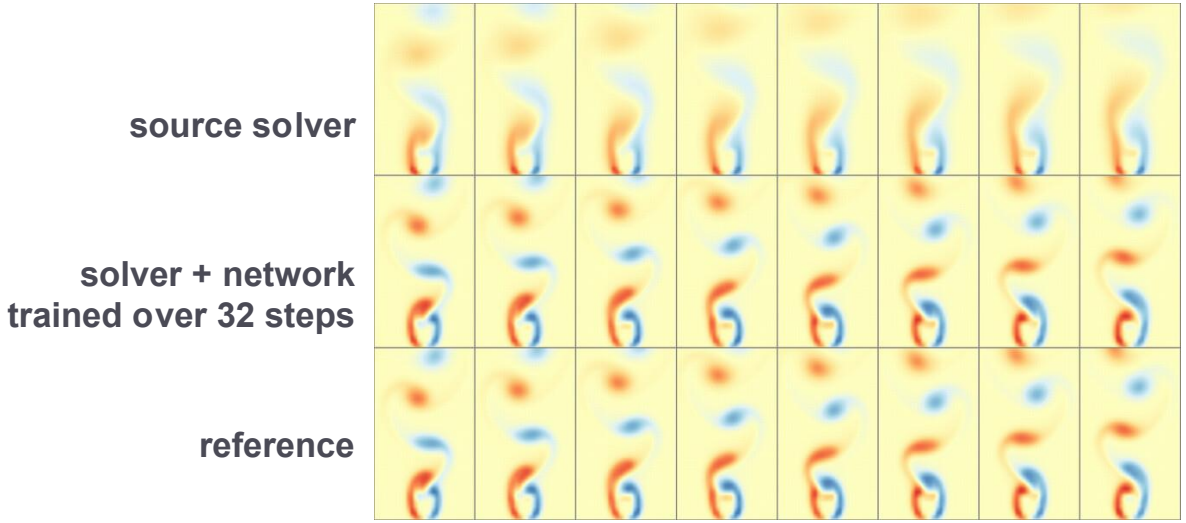


Figure: Um, Brand, Fei, Holl & Thuerey, Solver-in-the-Loop, NeurIPS 2020
Kochkov et al., PNAS 2021. Lam et al., Science 2023

Pushforward trick

- Rollout K − 1 steps without gradients, then train one step from the model's own prediction:

$$\mathcal{L}_0 = \left\| F_\theta\left(\mathrm{sg}[F_\theta^{K-1}(u_n)]\right) - u_{n+K} \right\|^2$$

- sg: stop gradient. The input contains the model's own errors; the target is the true u_{n+K}
- **Cost**: K forward passes and one backward pass; memory of one step
- **No gradient through the rollout**: the model does not learn how one step affects later ones
- MP-PDE unrolls at most 2 steps during training

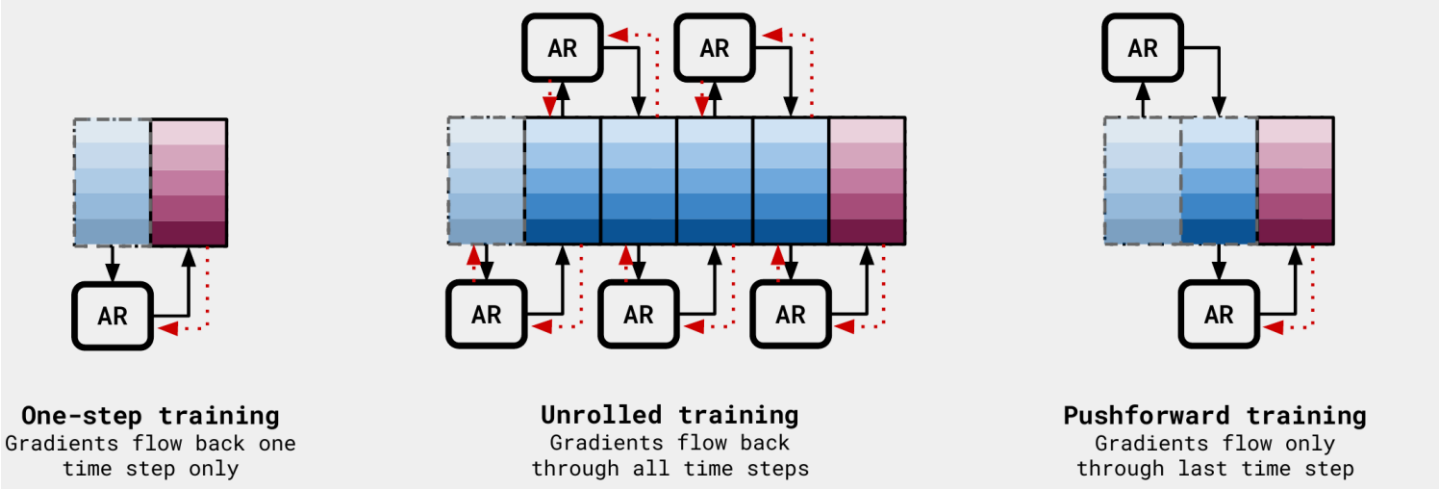


Figure: J. Brandstetter, D. Worrall & M. Welling, Message Passing Neural PDE Solvers, ICLR 2022. AR: one forward pass of the model; red: gradient flow

Comparison of training methods

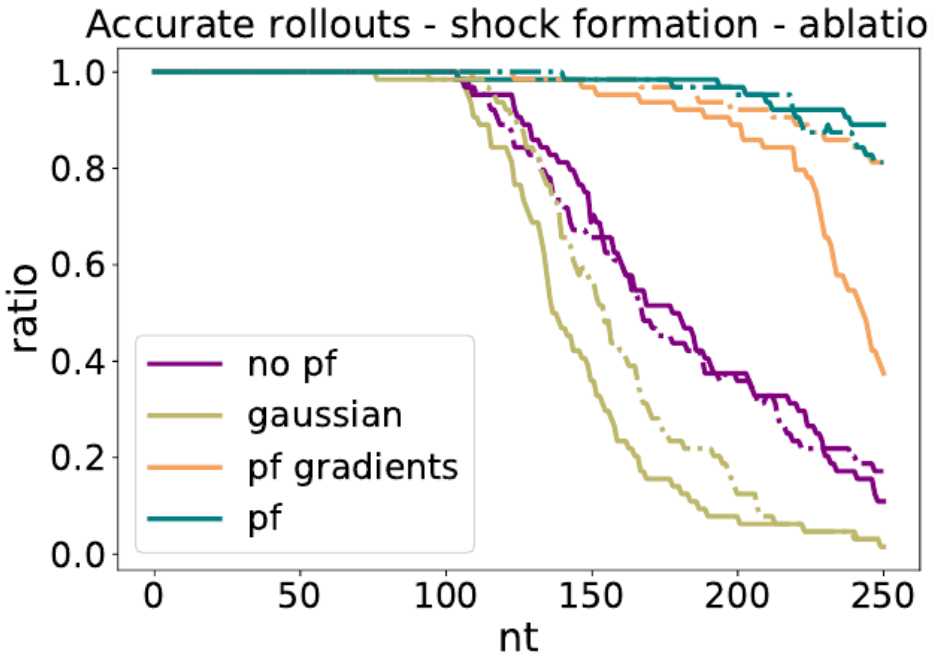
	One-step	Noise injection	Unrolled (K steps)	Pushforward (K steps)
Training input	true state	true state + noise	model's own states	model's own state
Gradient through the rollout	no	no	**yes**	no
Passes per sample	1 forward, 1 backward	1 forward, 1 backward	K forward, K backward	K forward, 1 backward
Memory	1 step	1 step	**K steps**	1 step
Limitation	drift, blow-up	noise unlike real errors	exploding gradients, memory	ignores multi-step effects
Reference	teacher forcing	Sanchez-Gonzalez et al. 2020	Um et al. 2020; Kochkov et al. 2021	Brandstetter et al. 2022

Train on the inputs the model will see in rollout

K: number of steps unrolled during training

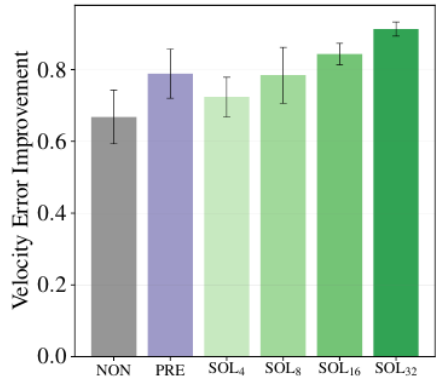
Pushforward and unrolled training results

Pushforward ablation (MP-PDE, 1D shock formation)

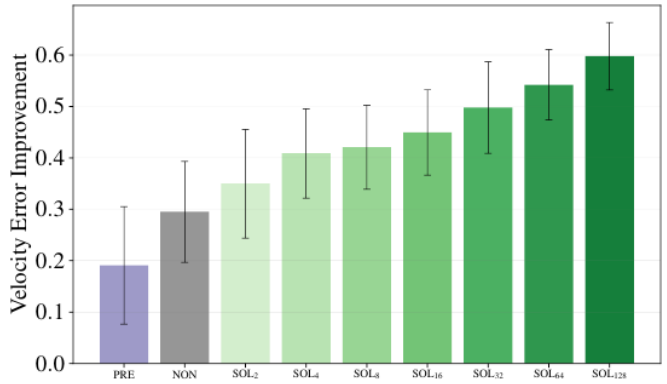


- y: fraction of test rollouts still accurate at step nt
- **pf** survives longest. **Gaussian** noise is worse than none here, and pf with gradients through the unroll is worse than pf

Unrolled training horizon (Solver-in-the-Loop)



(f) Unsteady wake, look-ahead



(g) Buoyancy-driven, look-ahead

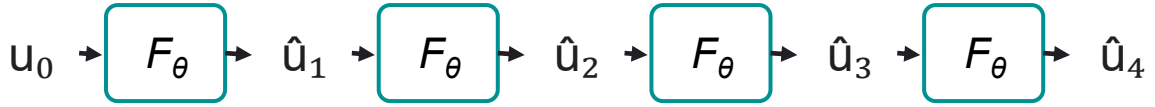
- y: reduction of the velocity error relative to the unaided solver (higher is better)
- SOL_n: trained by unrolling n steps through the solver. Longer look-ahead gives larger gains; NON and PRE are trained without the solver in the loop

Left: Brandstetter, Worrall & Welling, ICLR 2022. Right: Um, Brand, Fei, Holl & Thuerey, NeurIPS 2020.

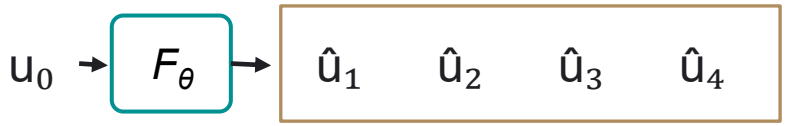
Temporal bundling and step size

- Error enters at every step and is amplified at every later step. For a fixed horizon, fewer steps means less of both
- **Larger Δt**: fewer steps but a harder map. The receptive field must cover the distance information travels in Δt
- **Temporal bundling**: predict the next K frames in one call, then rollout in blocks of K (Brandstetter et al. 2022)
- **Pangu-Weather**: four models with 1, 3, 6 and 24-hour steps. A 7-day forecast takes 7 steps of the 24-hour model instead of 168 one-hour steps (Bi et al., Nature 2023)

One frame per call: 4 calls, 4 chances to add and amplify error



Temporal bundling, K = 4: one call returns 4 frames



Pangu-Weather: Z500 error over 7 days, by step size of the model

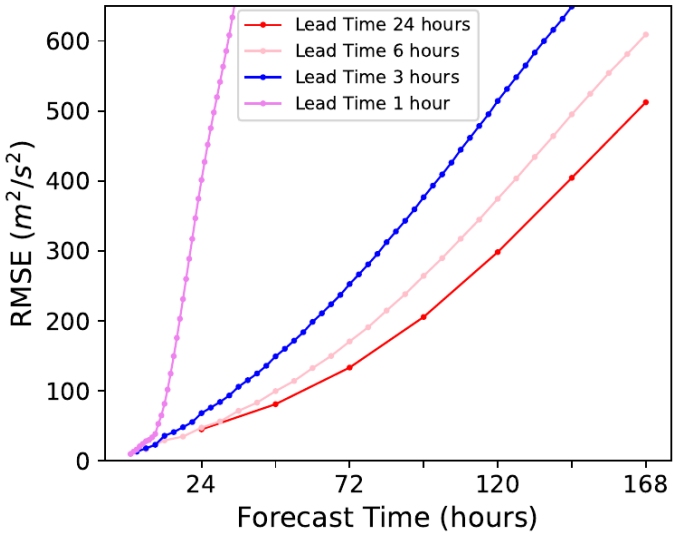
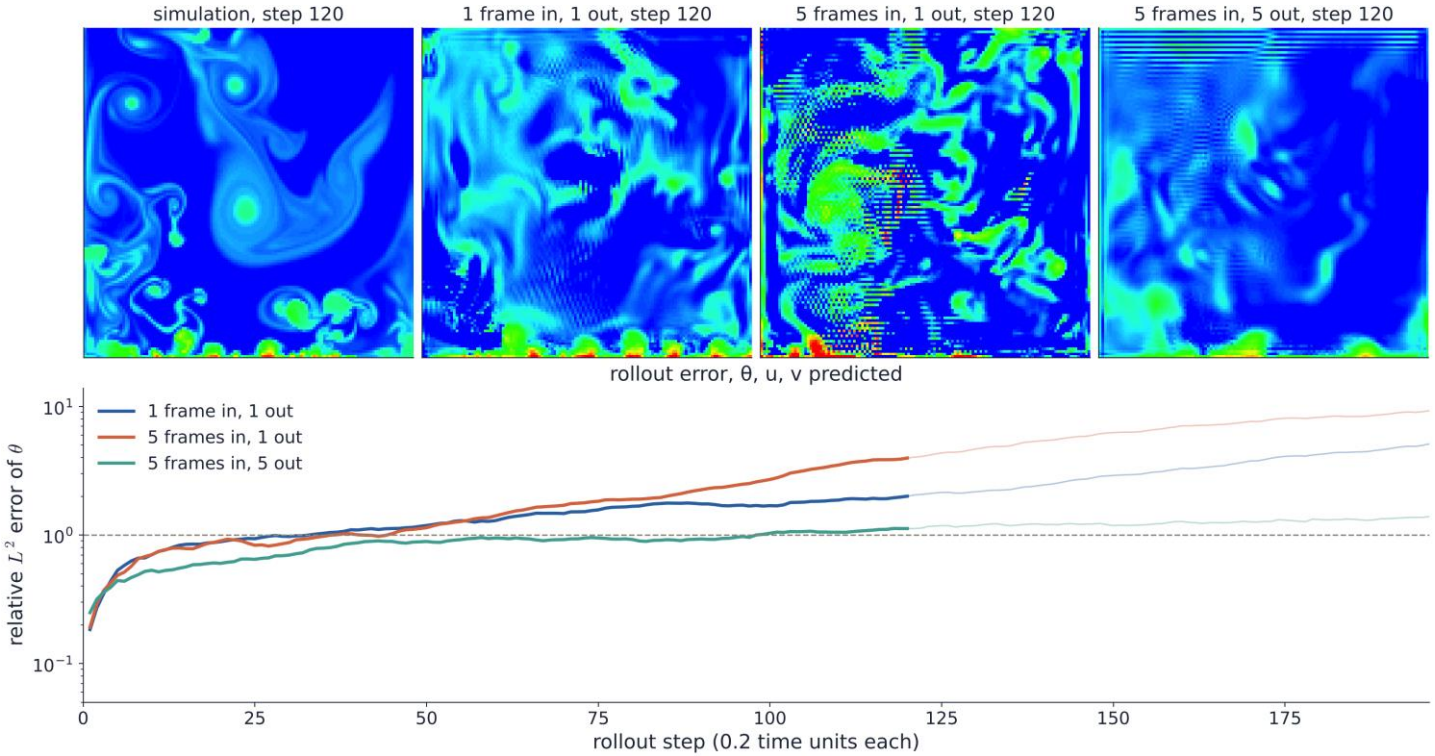


Figure: Bi et al., Pangu-Weather (Nature 2023). Temporal bundling: Brandstetter et al., ICLR 2022

Temporal bundling on BubbleML

- U-Net predicting θ, u and v, trained three ways: 1 frame in and 1 out; 5 in and 1 out; 5 in and 5 out
- 5 in, 5 out: the **highest** one-step error, but the **slowest** growth of rollout error. 200 frames take 40 model calls instead of 200
- 5 in, 1 out: extra input frames alone do not help; the error grows **faster** later
- Same finding on Kuramoto-Sivashinsky: more input frames lower the one-step error but shorten the accurate rollout (Lippe et al. 2023)



BubbleML_2, Twall = 106 ° C, 128 x 128, frames 0.2 apart; first 600 frames for training. U-Net from Lecture 6.
Kuramoto–Sivashinsky: Lippe et al., PDE-Refiner, NeurIPS 2023.

Other stabilization methods

- **Refinement**: PDE-Refiner predicts the next state, then refines it over several denoising steps with decreasing noise. Each step targets smaller, lower-amplitude components, including high k (Lippe et al., NeurIPS 2023)
- **Bound the growth**: if every layer has a bounded Lipschitz constant L, the model's J is bounded (e.g. spectral normalization, Miyato et al., ICLR 2018)
- **Correct a solver** instead of replacing it: the solver keeps its stability, the network corrects its error
- **Sample, not average**: generative models avoid MSE blur
- **Continuous time**: neural ODEs

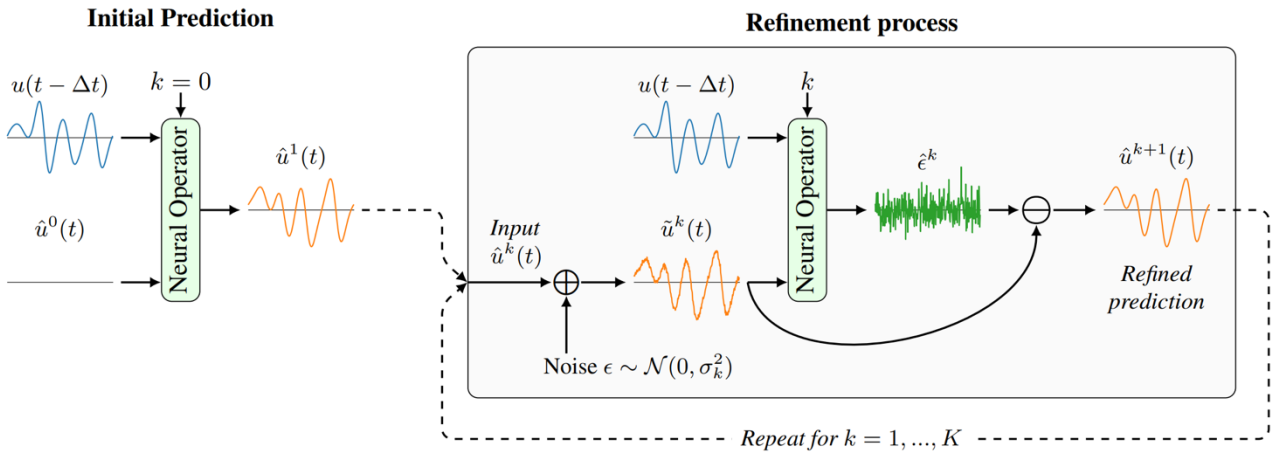


Figure: P. Lippe, B. Veeling, P. Perdikaris, R. Turner & J. Brandstetter, PDE-Refiner: Achieving Accurate Long Rollouts with Neural PDE Solvers, NeurIPS 2023

Summary

- An autoregressive model is applied to its own output; a lower one-step error does not guarantee a better rollout
- Rollout error grows through the one-step error δ *and* the amplification J: bounded, linear or exponential
- In published models, errors grow first at high wavenumbers, where the data has little energy
- Rollout inputs drift away from the training inputs (distribution shift); MSE predicts the mean, so rollouts blur
- Train on the model's own states: noise injection, unrolled training, pushforward trick
- Take fewer, larger steps: temporal bundling, multiple step sizes. Extra input frames alone do not help

Project deliverables and deadlines

Milestone	Due	What to submit
Project abstract	**Thu, Oct 22**	One page: solo or team (2-3 students), the problem, data, baselines, metrics, plan, and GPU estimate
Mid-project check-in	**Thu, Nov 12**	Max 5 slides presentation update: baselines reproduced, first results, revised plan
Final presentation	**Tue, Dec 8**, 8:00-10:00 AM	In-class presentation by each team in the final exam slot
Final report	**Fri, Dec 11**	Max 6 pages conference-style paper, with code and a README that reproduces the main results

- The project is **75% of the grade**
- Compute: HPC3 GPUs. Plan for a single GPU per run
- Four project types: reproduce and test a recent paper, extend a method, compare methods on a scientific dataset, or apply ML to data from your own research

Project scope and evaluation

- Research-scale data, published models, an open problem
- A problem with a measurable outcome, stated in the abstract
- Strong baselines: published models with public code, reproduced to their reported accuracy
- Metrics: rollout error over time, energy spectra, conserved quantities, physical statistics such as heat flux
- Controlled comparisons: same data split, model size and GPU budget across methods; several seeds
- Report negative results: a method that does not help, with the analysis, is a meaningful result

Public datasets

- Course data: heat equation, BubbleML
- PDEBench: 1D and 2D PDEs
- PDEArena: Navier-Stokes, shallow water
- The Well: 16 physics datasets
- MeshGraphNets: flows on meshes
- MD17: molecular dynamics
- WeatherBench 2: global weather

Define the problem, baselines and metrics before writing code

PDE-Refiner beyond Kuramoto-Sivashinsky

- **Paper**: PDE-Refiner (Lippe et al., NeurIPS 2023) refines each prediction over a few denoising steps. On Kuramoto-Sivashinsky, its accurate rollout time is longer than that of an MSE-trained U-Net
- **Reproduce**: train the MSE U-Net and PDE-Refiner on the KS data with the public code; match the reported accurate rollout times
- **Tests**
  - Equal compute: PDE-Refiner uses several network evaluations per step. Compare with an MSE model of the same cost and with an ensemble
  - New system: a 2D dataset from The Well with a different energy spectrum
  - Ablations: number of refinement steps, minimum noise level
- **Outcome**: conditions under which refinement improves the rollout, with energy spectra of the errors

Lippe, Veeling, Perdikaris, Turner & Brandstetter, PDE-Refiner, NeurIPS 2023; code in github.com/microsoft/pdearena. The Well: Ohana et al., NeurIPS 2024

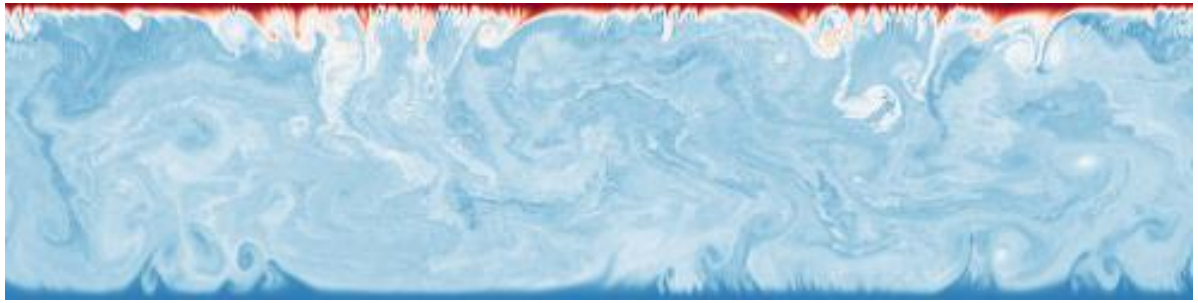
Training methods for long rollouts of boiling

- **Methods**: noise injection, pushforward and unrolled training, and temporal bundling (this lecture)
- **Extension**: apply them to multiphase flow, where the state includes temperature, velocity and a moving interface; design and test one modification
- **Data**: BubbleML 2 pool boiling at full resolution, several wall temperatures; held-out wall temperatures for testing
- **Plan**
  - Baseline: U-Net or FNO with temporal bundling that predicts temperature, velocity and interface, trained on one step
  - Add pushforward and unrolled training, then the modification, for example a loss weighted near the interface
  - Metrics: stable rollout length, heat flux at the heater, bubble departure frequency, energy spectra
- **Outcome**: comparison of the training methods on long-rollout boiling statistics and on training cost

BubbleML: Hassan et al., NeurIPS 2023 (BubbleML_2 on Hugging Face). Pushforward and temporal bundling: Brandstetter, Worrall & Welling, ICLR 2022

U-Net, FNO and transformer on Rayleigh-Bénard convection

- **Problem**: compare a U-Net, an FNO and a transformer on 2D convection at a fixed GPU budget
- **Data**: Rayleigh-Bénard convection from The Well, with its data loaders and baseline models



- **Comparisons**
  - One-step and rollout error, at equal parameter count and equal training time
  - Data scaling: error against the number of training trajectories
  - Resolution: train at one resolution, test at another
  - Physics: Nusselt number and energy spectra over long rollouts
- **Outcome**: ranking of the three models and how it changes with data size, rollout length and resolution

The Well: Ohana et al., NeurIPS 2024 (polymathic-ai.org/the_well). FNO: Li et al., ICLR 2021. U-Net: Ronneberger et al., MICCAI 2015

Surrogate model for your own research data

- **Data**: simulations or measurements from your own research
- **Example**: temperature of a processor chip under a changing workload, from a thermal simulator
  - Task: predict the chip temperature map over time from the power map of each core, for workloads not in the training set
  - Baselines: the thermal simulator (accuracy and run time) and a published operator model (FNO or U-Net)
  - Metrics: peak temperature and hotspot location over long rollouts, speedup over the simulator
- **Requirements**: data must be sharable to check results, enough independent runs for a held-out test set, and a physical metric in addition to pixel error
- **Outcome**: accuracy and speed of the surrogate relative to the simulator, and the conditions under which it fails

After class

- Brandstetter, Worrall & Welling, **Message Passing Neural PDE Solvers**, ICLR 2022: pushforward trick and temporal bundling
- Sanchez-Gonzalez et al., **Learning to Simulate Complex Physics with Graph Networks**, ICML 2020: noise injection
- Lippe et al., **PDE-Refiner: Achieving Accurate Long Rollouts with Neural PDE Solvers**, NeurIPS 2023
- McCabe, Harrington, Subramanian & Brown, **Towards Stability of Autoregressive Neural Operators**, TMLR 2023
- Ross & Bagnell, **Efficient Reductions for Imitation Learning**, AISTATS 2010: why errors compound

Next, Lecture 8 (Tue, Oct 20): Recurrent networks and neural ODEs

NEXT: HANDS-ON

From concepts to code

→ **Open the Lecture 7 notebook**

1. Predict θ, u and v; compare the rollout with simulated velocity and with predicted velocity

2. Retrain with the pushforward trick and with noise injection; compare rollout error and energy spectra with one-step training