CRITEO AI Lab

SORBONNE UNIVERSITÉ

CEA-EDF-INRIA Numerical Analysis Summer School 2025
Solving PDE in fields physics faster with physics-based machine learning

Neural Surrogate Models of Physical Dynamics: Learning to Generalize:

Patrick Gallinari
Sorbonne Universite & Criteo AI Lab - Paris

OoD generalization and physical phenomena

Explicit models

- Physical models
  - Developed from first principles
  - Rely on a deep understanding of the physical phenomenon
  - Capture some form of « causality » between well identified physical variables
  - Generalize to different contexts

Implicit models

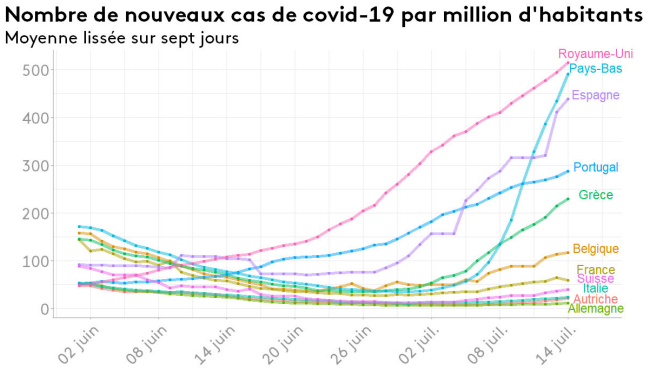
- Agnostic statistical models
  - Learn from data
  - Capture correlations between data features and not between explicit physical variables
  - Does not generalize outside the training distribution

- OoD problem in physics: NNs do not generalize
- Generalization is critical for the adoption of NNs in physics
- Physics complexity in often order of magnitude larger than in classical ML settings
- Cannot be solved with classical ERM

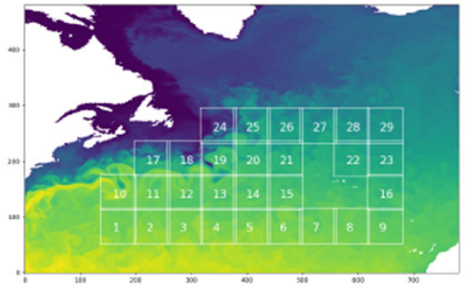
OoD generalization, modeling physical phenomena, context and examples

- Assumption: one underlying phenomenon - different environments

Modelling epidemics in different countries

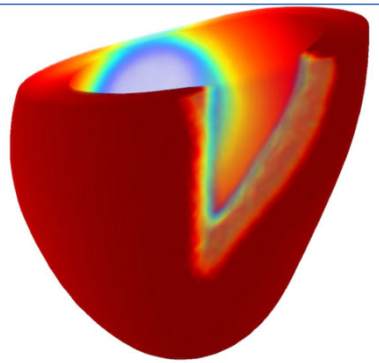


Predictions of sea surface temperature from satellite data, Fig. Pajot et al. 2018



Sub regions extracted for the dataset. Test regions are regions 17 to 20.

Modeling electrophysiology of cardiac tissue from different patients, Fig. Fresca et al. 2020



- Complexity of the modeling problem
- Focus of the presentation: solving parametric PDEs

OoD Generalization: what could be learn from ML?

OoD generalization: Infer causality, improve robustness

OoD adaptation: meta-learning

OoD Generalization in Machine Learning

- M. Arjovski et al 2020 – OoD generalization problem in ML

Slides: L. Bottou invited talk ICLR 2019

Nature does not shuffle the data. We do!

We collect data
- at different points in time and in space
- in different experimental settings
- with different biases

} environments

Then we shuffle the records and pretend that they are independent and identically distributed

Nature does not shuffle the data. We do!

We collect data
- at different points in time and in space
- in different experimental setti
- with different bi

Then we shuf
they are

Shuffling the data is a loss of information.

nd that
nd identically distributed

Invariant Risk Minimization

Get rid of spurious correlations

Leverage the different environments to learn invariant representation

Objective: extrapolate outside the training domain

OoD Generalization in Machine Learning

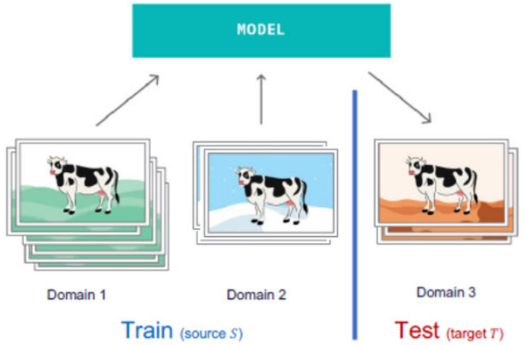
Out of distribution (OoD) generalization

How to train models that perform will outside their training distribution

Includes a variety of problems/ approaches

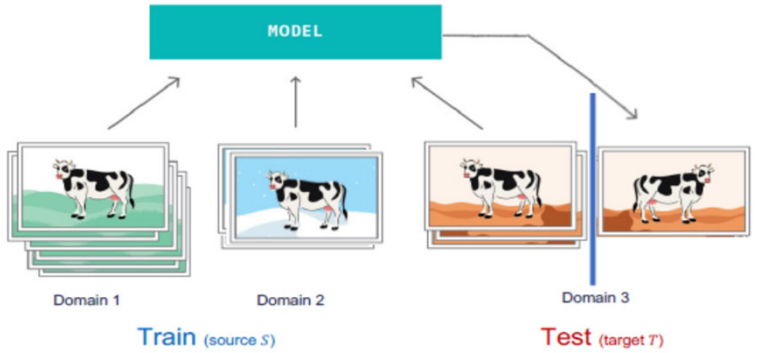
0-shot generalization

- Learn from several environments
- Learn environment invariant representations
- 0-shot: do not use new environment data



Few-shot adaptation

- Learn from several environments
- Learn environment-conditionned models
- Few-shot adaptation: with scarce data from a new environment



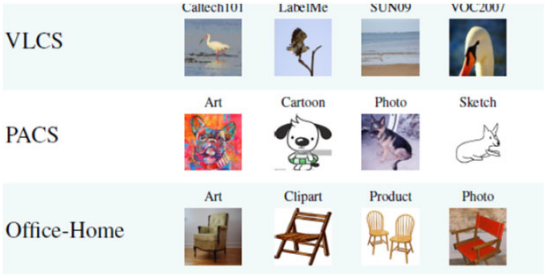
OoD Generalization in Machine Learning

Training set of environments : $D_{Train} = \{D_1, \dots, D_N\}$

Leverage environment information

0-shot generalization

- Approaches
  - Robust estimation (DRO) (Sagawa et al. 2020)
  - Invariant risk minimization (IRM) (Arjovsky et al. 2020)
  - Risk Extrapolation (REx) (Krueger et al. 2021)
- DomainBed benchmark (Gulrajani et al 2021)



- ERM is as good as any of these strategies ☹

Few-shots adaptation

- Approaches
  - e.g. meta-learning MAML (Finn et al 2017)

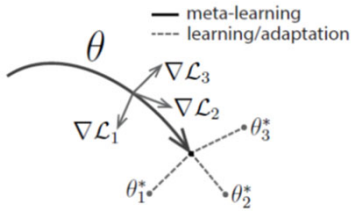


Figure 1. Diagram of our model-agnostic meta-learning algorithm (MAML), which optimizes for a representation θ that can quickly adapt to new tasks.

  - Learn shared parameters θ from several environments. Adapt $\theta_e = \theta + \delta\theta_e$ for environment D_e
  - Complexity: double optimisation process

Tackling the generalization problem for physical dynamics

✓

Generalization for physical dynamics

Learn a model for physical dynamics that generalizes to different environments

- Data often come from the numerical simulation of PDEs
- An environment is characterized by:
  - Physics, e.g. multiple PDEs
  - Parameters for a given PDE
  - Initial value distributions, boundary conditions, etc

- Data-driven models for spatio-temporal forecasting
  - PDE is unknown
  - Objective: learn forecast models from trajectories

Tackling the generalization problem for physical dynamics
Solving parametric PDEs

We consider a class of spatio-temporal partial differential equations dependent on **parameters**
How to solve parametric equations with data-driven approache?

PDE coefficients
Forcing terms

Explicit modeling

$$\frac{\partial u}{\partial t} = g(c, f; x, t, u, \nabla u, \nabla^2 u, \dots)$$

Initial condition

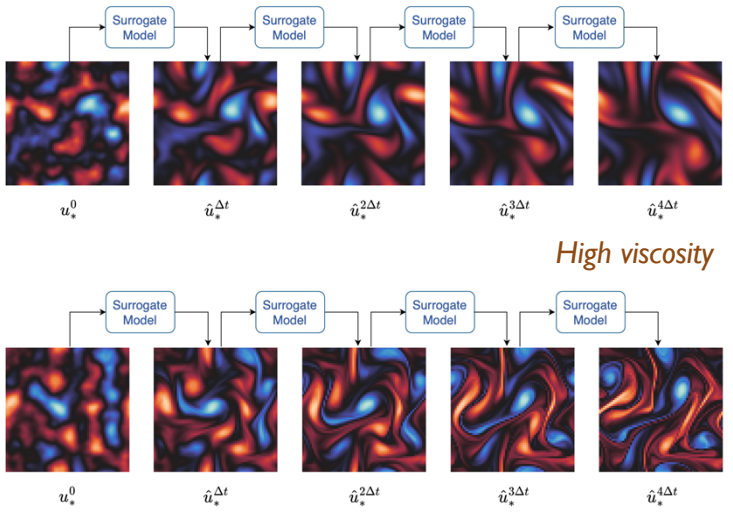
$$u(x, t = 0) = u^0(x)$$

$$B(b; u, \nabla u, x, t) = 0 \qquad \forall (x, t) \in \partial\Omega \times \mathbb{R}_*^+$$

Boundary type

Implicit modeling

Neural Surrogates should generalize to changes in new initial conditions and PDE parameters



Generalization for dynamical systems
Spatio-temporal forecasting – data-driven approches – ERM approaches

- Training: sample environment parameters / sample trajectories from each environment

- Graph Neural Networks

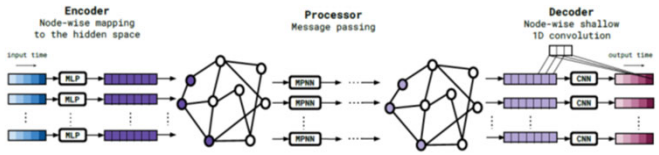


Fig. Brandstetter et al. 2022

- Neural operators

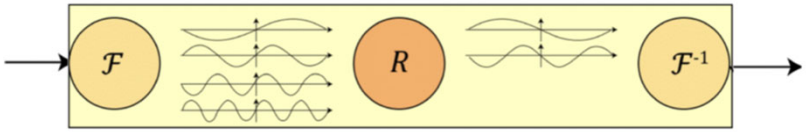


Fig. Li et al. 2021

- Foundation models

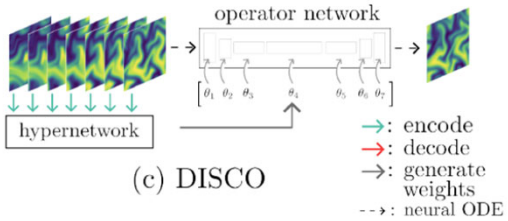


Fig. Morel et al. 2025

- Limited OoD generalization / extrapolation abilities

Generalization for dynamical systems
Spatio-temporal forecasting – data-driven approches

- OoD generalization
  - Requires conditioning on characteristics of the new domain
    - PDE parameters, e.g. coefficients: prior knowledge

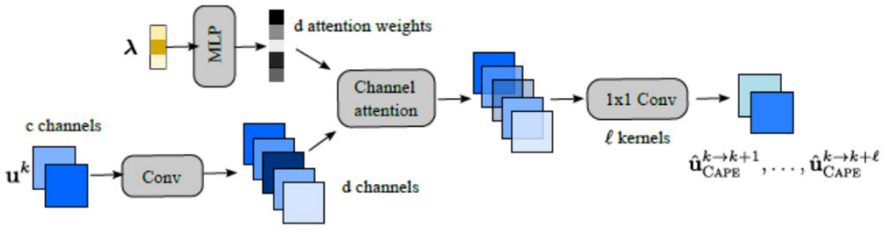


Figure 2. The CAPE module for one type of convolution (residual connections are omitted).

Fig. Takamoto et al. 2023
Learning Neural PDE Solvers with Parameter-Guided Channel Attention

    - Fine tuning: requires significant amount of data

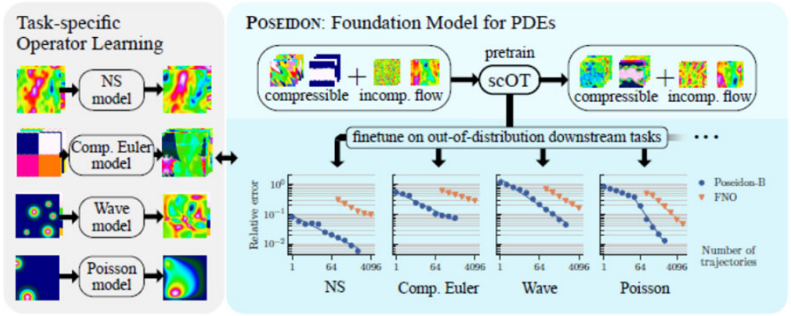


Fig. Herde et al. 2024
Poseidon: Efficient Foundation Models for PDEs

Tackling the generalization problem for dynamical systems: Conditional Adaptation

Tackling the generalization problem for dynamical systems

- Focus: solving parametric PDEs
- Assumption
  - Data come a set of domains $E = \{e^i\}$
    - All the domains share the same form of the dynamics modeled by a PDE
    - One domain corresponds to a spoecific PDE instance
      - What differs across PDE instances: parameters of the dynamics
- Problem : extrapolation
  - Learn on a sample of the domains' distribution – set of PDEs
  - Generalize on new domains not seen during training- e.g. new PDE instances
- Methods: few shot adaptation
  - How?

Tackling the generalization problem for dynamical systems

- One particular example of parametric PDEs experimental setting
  - Combined equation (Brandstetter 2022)
    - Parameters: Coefficients $\theta = (\alpha, \beta, \gamma)$
      - Heat equation $\theta = (0, \eta, 0)$
      - Burgers $\theta = (0.5, \eta, 0)$
      - KdV (Korteweg–De Vries) $\theta = (3, 0{,}1)0{,}1]$

$$[\partial_t u + \partial_x(\alpha u^2 - \beta \partial_x u + \gamma \partial_{xx} u)](t, x) = \delta(t, x),$$

$$\delta(t, x) = 0, \quad u_0(x) = \sum_{j=1}^{J} A_j \sin(2\pi \ell_j x / L + \phi_j).$$

  - Objective: train from a finite sample of the parameters, generalise to new samples
  - Training
    - (α, β, γ) sampled in the range $([0{,}1], [0, 0.04], [0{,}1])$ – each sample corresponds to a PDE instance
  - Testing
    - In-distribution: trajectories from the same parameter range
    - Out-distribution: trajectories outside the training parameter range

Tackling the generalization problem for dynamical systems

- Training

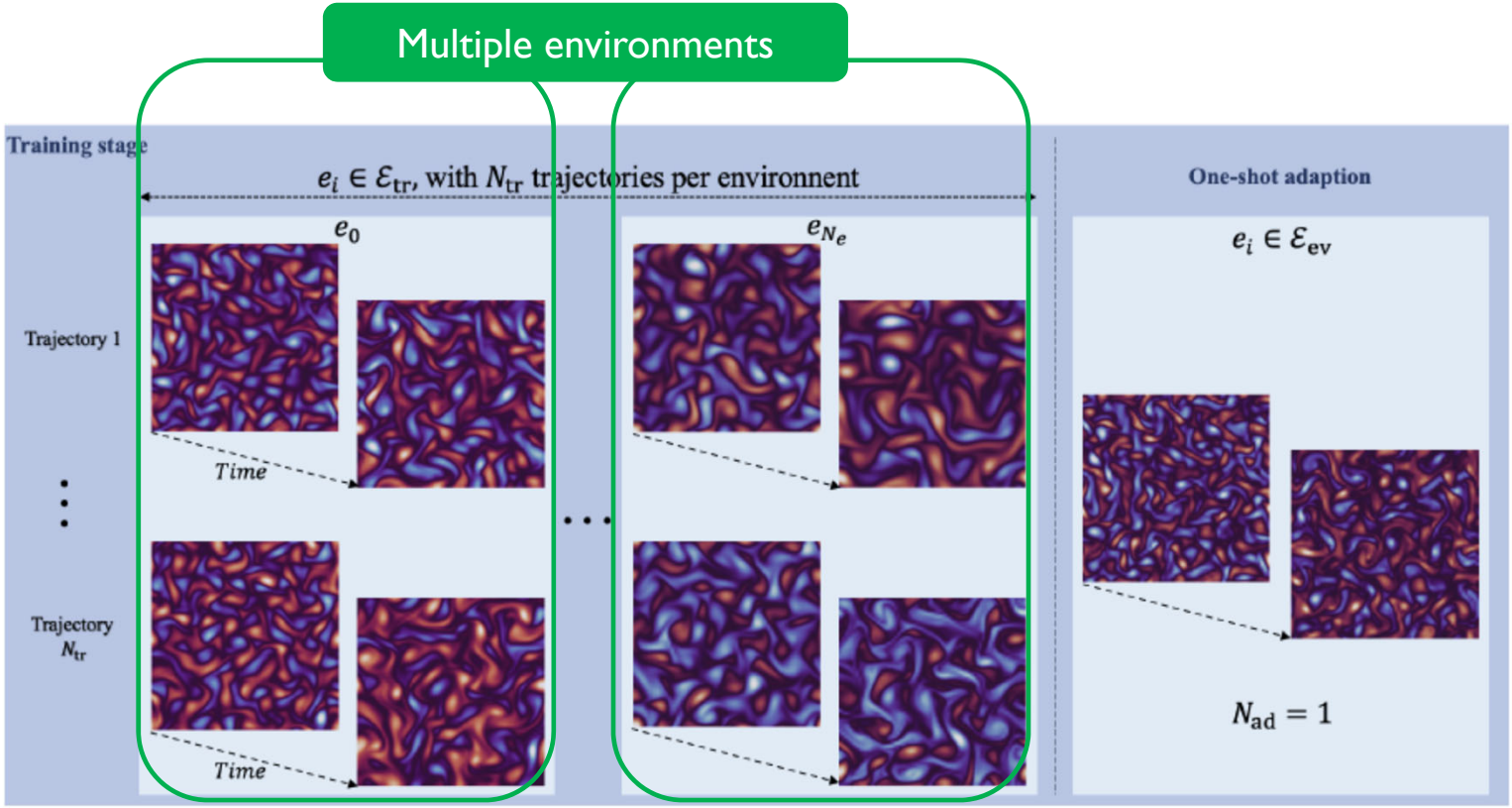


Figure 1: Multi-environment setup for the Kolmogorov PDE. The model is trained on multiple environments with several trajectories per environment (left). At inference, for a new unseen environment it is adapted on one trajectory (right).

Tackling the generalization problem for dynamical systems
CODA & GEPS frameworks (Kirchmeyer et al. 2022, Kassai et al. 2024)

- Training

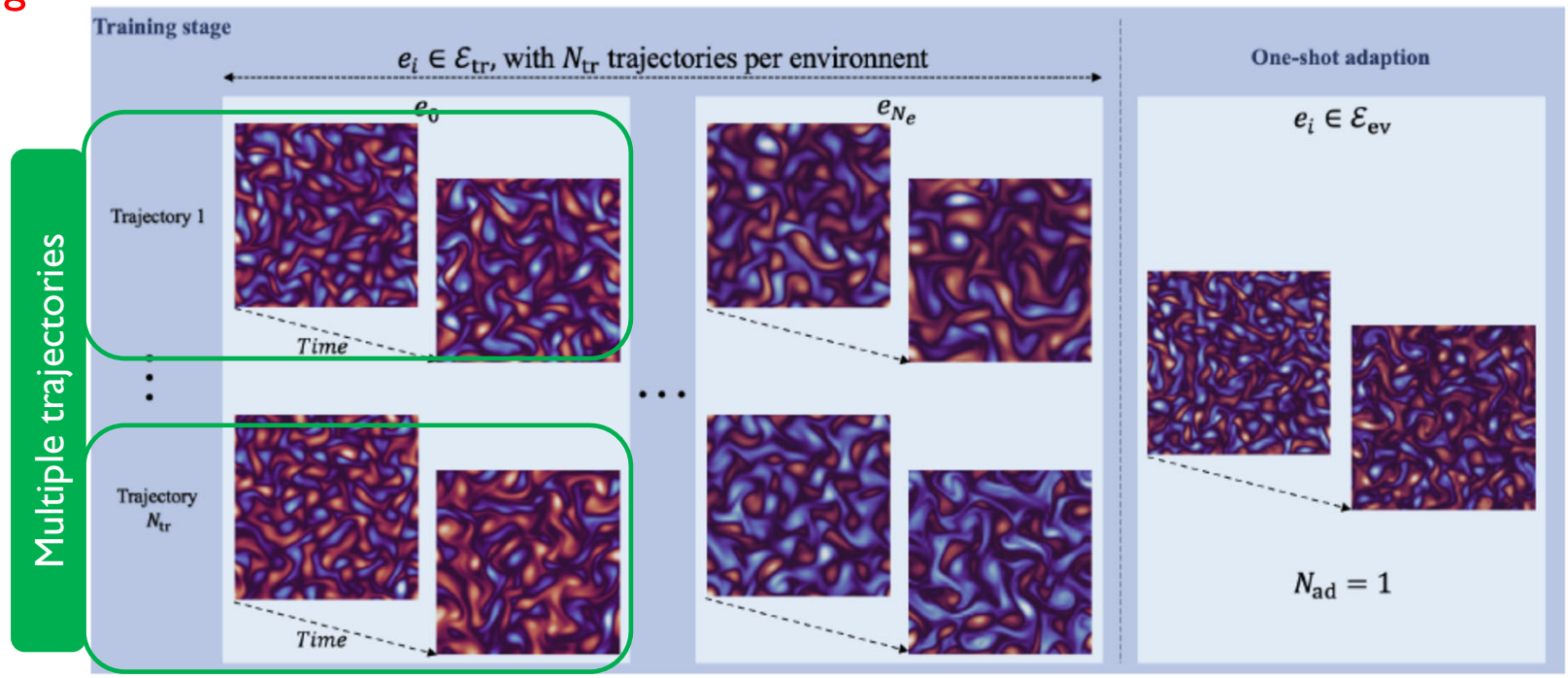


Figure 1: Multi-environment setup for the Kolmogorov PDE. The model is trained on multiple environments with several trajectories per environment (left). At inference, for a new unseen environment it is adapted on one trajectory (right).

Tackling the generalization problem for dynamical systems

- Adaptation

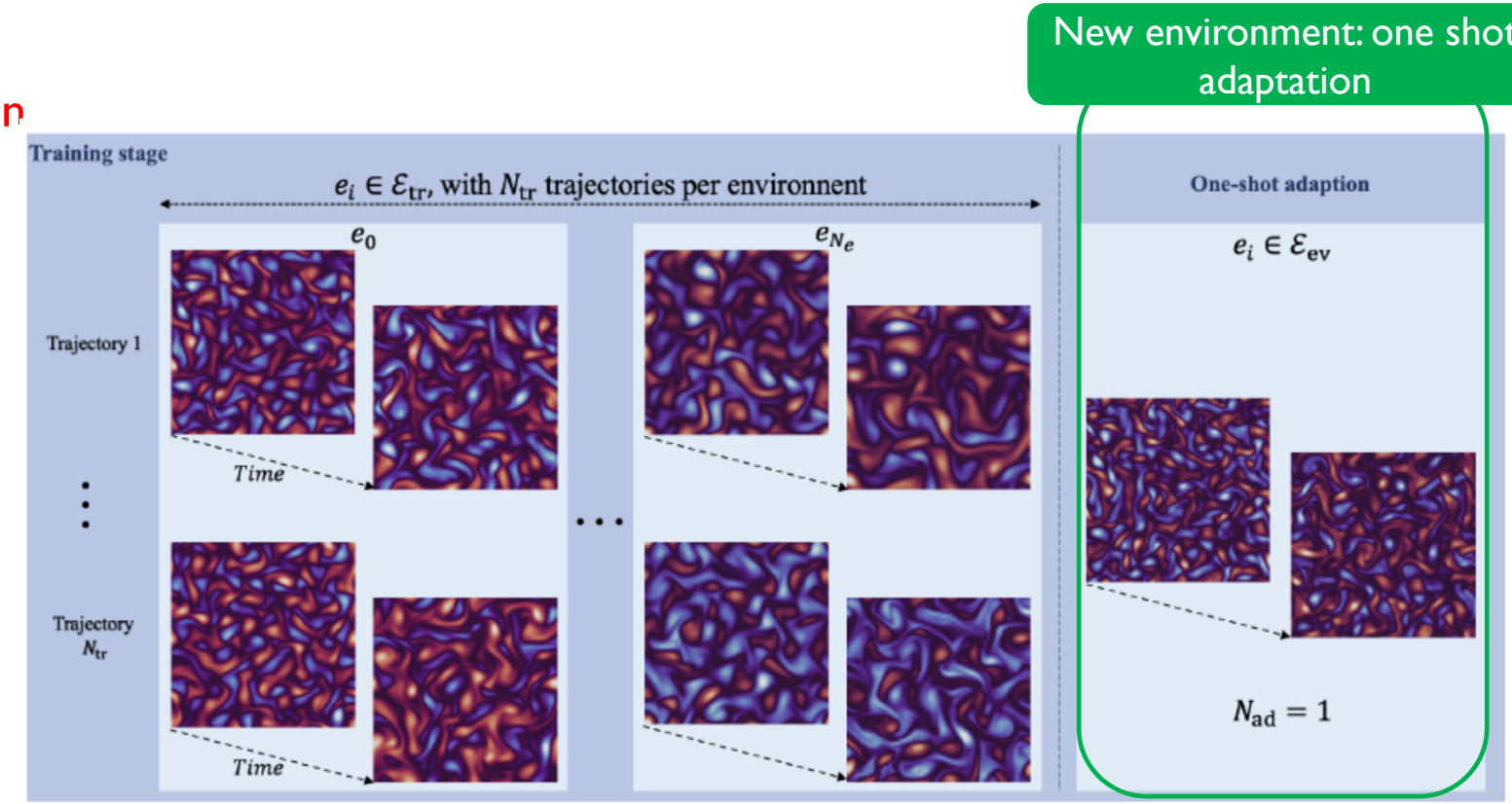
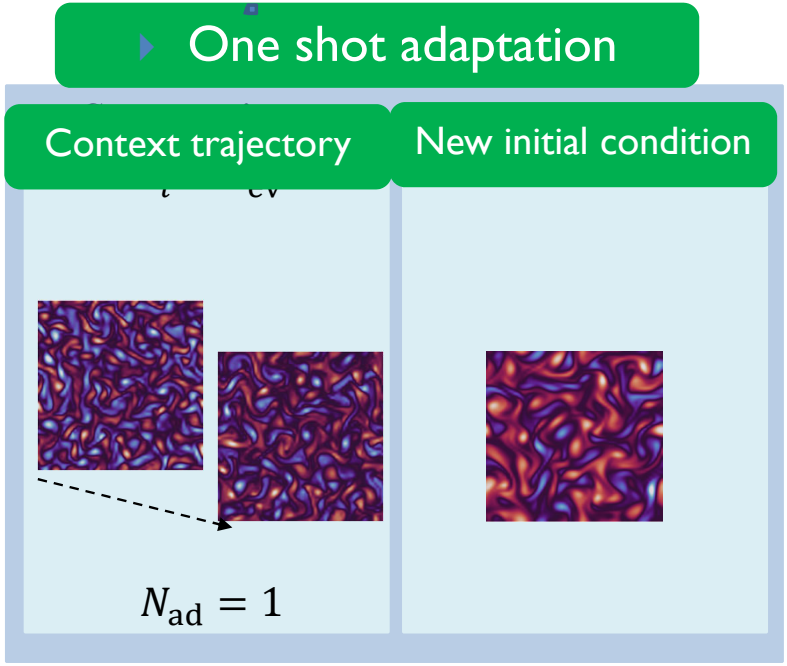


Figure 1: Multi-environment setup for the Kolmogorov PDE. The model is trained on multiple environments with several trajectories per environment (left). At inference, for a new unseen environment it is adapted on one trajectory (right).

Tackling the generalization problem for dynamical systems

- Inference on a new instance of the PDE
  - Fast adaptation through few shot learning
  - Given a sample of the new domain, adapt fast to solve IVPs on this domain



Tackling the generalization problem for dynamical systems:
Meta learning - Gradient based approaches

Kirchmeyer et al., CODA: Generalizing to new Physical Systems via Context-Informed Dynamics Models, ICML 2022

Kassai et al., GEPS: Boosting Generalization in Parametric PDE Neural Solvers through Adaptive Conditioning, Neurips 2024

Tackling the generalization problem for dynamical systems (GEPS, Kassai et al. 2024)

ERM baselines vs adaptive conditioning
as a function of the number of training environments
In distribution - Initial Value Problem

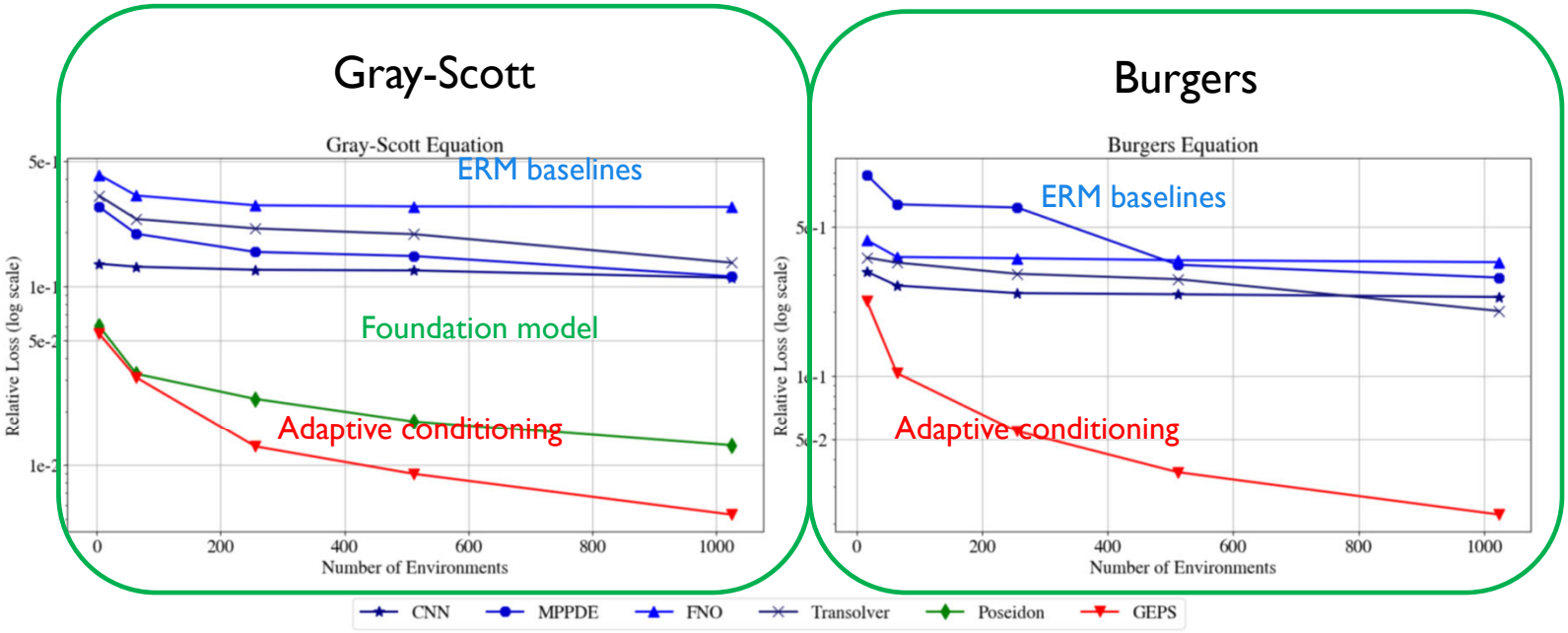


Figure 2: Comparison of ERM approaches (shades of blue) and Poseidon foundation model (green) with our framework GEPS (red) when increasing the number of training environments.

Tackling the generalization problem for dynamical systems (GEPS, Kassai et al. 2024)

ERM baselines vs adaptive conditioning
as a function of the number of training trajectories/ environment
Initial Value Problem

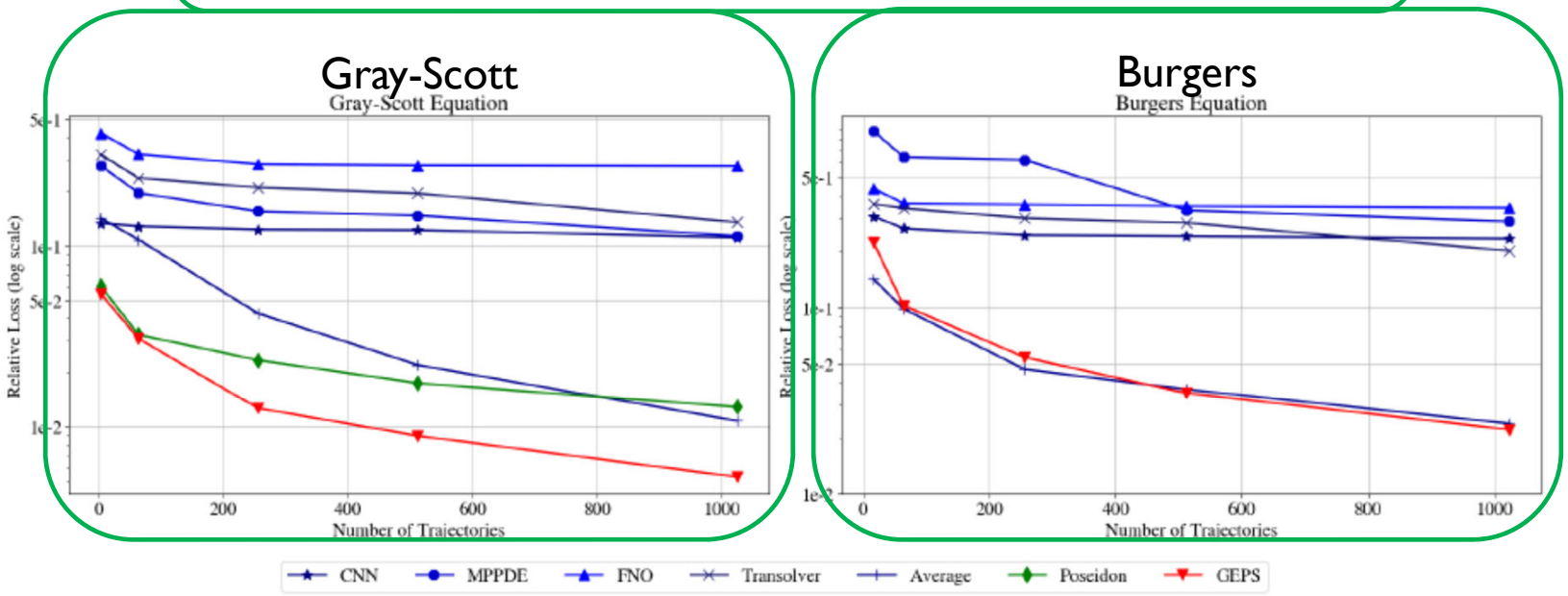


Figure 3: Comparison of ERM approaches (shades of blue) and Poseidon (green) with our framework GEPS (red) when increasing the number of trajectories per environment.

Tackling the generalization problem for dynamical systems (GEPS, Kassai et al. 2024))

ERM baselines vs adaptive conditioning

Out of Distribution - IVP

Fine tuning or Adapting on one trajectory for the new environment

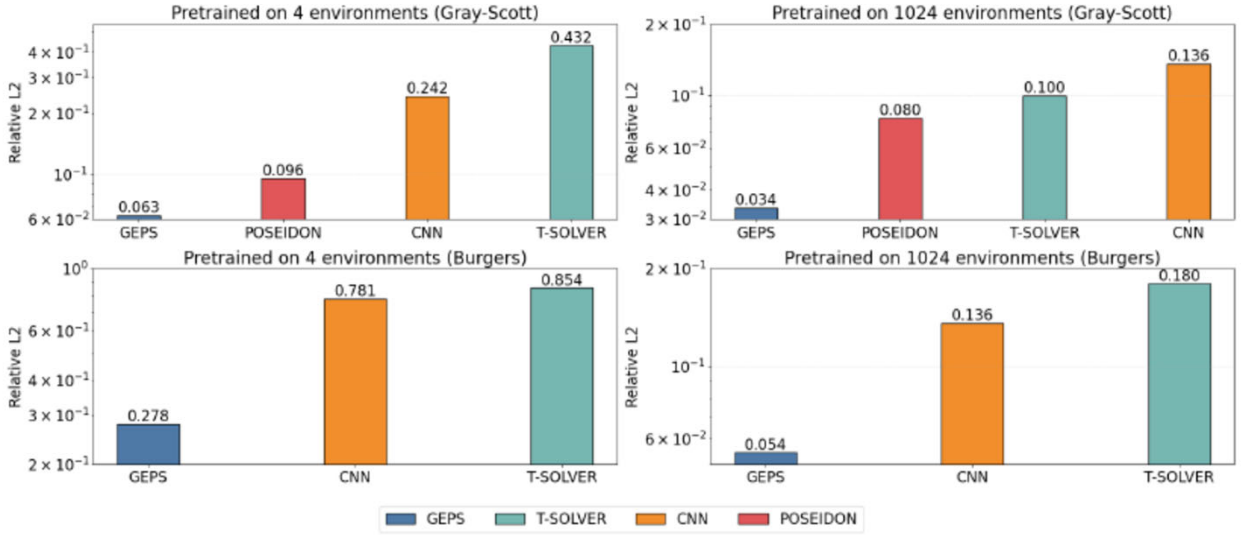


Figure 4: Out-distribution generalization on 4 new environments using one trajectory per environment for fine-tuning or adaptation. Models have either been pretrained on 4 environments (left column) or 1024 environments (right columns). Metric is Relative L2 loss.

Tackling the generalization problem for dynamical systems (GEPS, Kassai et al. 2024)

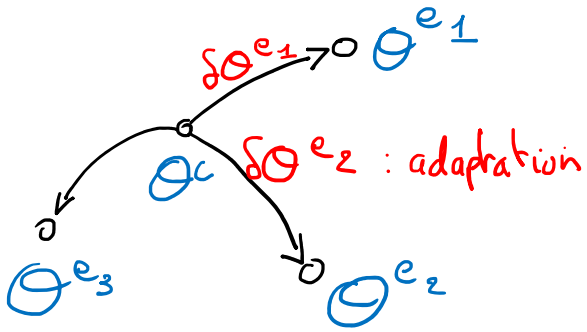
ERM baselines vs adaptive conditioning
In and Out of distribution
Context: historical data

Table 1: In-distribution and out-distribution results comparing different history window sizes. Metric is the Relative L2 loss.

History →	3		5		10	
Method	*In-d*	*Out-d*	*In-d*	*Out-d*	*In-d*	*Out-d*
Burgers equation						
Transolver	1.95e-1	3.22e-1	1.12e-1	3.03e-1	5.64e-2	2.49e-1
FNO	4.28e-1	7.68e-1	3.07e-1	6.43e-1	6.13e-2	2.55e-1
CNN	1.84e-1	4.44e-1	1.62e-1	3.34e-1	3.16e-2	6.32e-2
GEPS	**1.25e-2**	**1.63e-2**	**8.61e-3**	**1.04e-3**	**6.14e-3**	**8.73e-3**
Gray-Scott equation						
Transolver	1.82e-1	4.33e-1	9.88e-2	3.90e-1	9.57e-2	3.60e-1
FNO	1.86e-1	4.67e-1	1.76e-1	3.87e-1	1.93e-1	4.03e-1
CNN	7.12e-2	3.51e-1	5.96e-2	2.18e-1	6.54e-2	2.23e-1
GEPS	**4.02e-2**	**5.78e-2**	**3.04e-2**	**5.02e-2**	**3.82e-2**	**5.14e-2**

Tackling the generalization problem for dynamical systems
CODA framework (Kirchmeyer et al. 2022)

- How to: Intuition
  - Learn to condition the learned function f_{θ^e} on the environment e
  - So that it could adapt fast and with a few samples to a new environment
  - Adaptation rule: $\theta^e = \theta^c + \delta\theta^e$
    - θ^c shared parameters trained on a sample of the environment distribution
      - i.e. on a set of trajectories sampled for sampled from a set of environments
    - $\delta\theta^e$ environment specific parameters infered for each new environment



- Under two key constraints
  - $C1$: Locality constraint for small weight deviation $\delta\theta^e$: fast convergence
  - $C2$: Low Rank Adaptation: adaptation performed in a small dimensional parameter subspace: for sample and parameter efficiency

Tackling the generalization problem for dynamical systems
CODA framework (Kirchmeyer et al. 2022)

- Dynamical function for domain e
  - $\boxed{f_{\theta^e} \; with \;\; \theta^e = \theta^c + \delta\theta^e}$
- Training objective
  - $\boxed{\min_{\theta^c, \delta\theta^e; e \in E} \sum_{e \in D_{Train}} \|\delta\theta^e\|^2 \;\; \text{s.t. } \forall x^e \in D^e, \forall t, \frac{dx^e(t)}{dt} = f_{\theta^c + \delta\theta^e}\left(x^e(t)\right)}$
    - $C1$: Locality constraint $\min_e \|\delta\theta^e\|^2$: θ^e should lie in the neighborhood of θ^c
    - $C2$: Low rank adaptation - Small intrinsic dimensionality for $\delta\theta^e$
      - Implemented by learning a code specific to each environment + hypernetwork
        - $\delta\theta^e$ generated via a hypernetwork: $\boxed{\delta\theta^e = W\xi^e}$
          - Compact code ξ^e infered for each domain through auto-decoding, W are shared parameters

Tackling the generalization problem for dynamical systems
CODA framework (Kirchmeyer et al. 2022)

- Training objective

$$\min_{\theta^c, W, \xi^e; e \in E} \sum_{e \in D_{Train}} \|W\xi^e\|^2 \text{ s.t. } \forall x^e \in D^e, \forall t, \frac{dx^e(t)}{dt} = f_{\theta^c + W\xi^e}\left(x^e(t)\right)$$

  - In green: shared parameters across the environments
  - In red: environment specific parameters

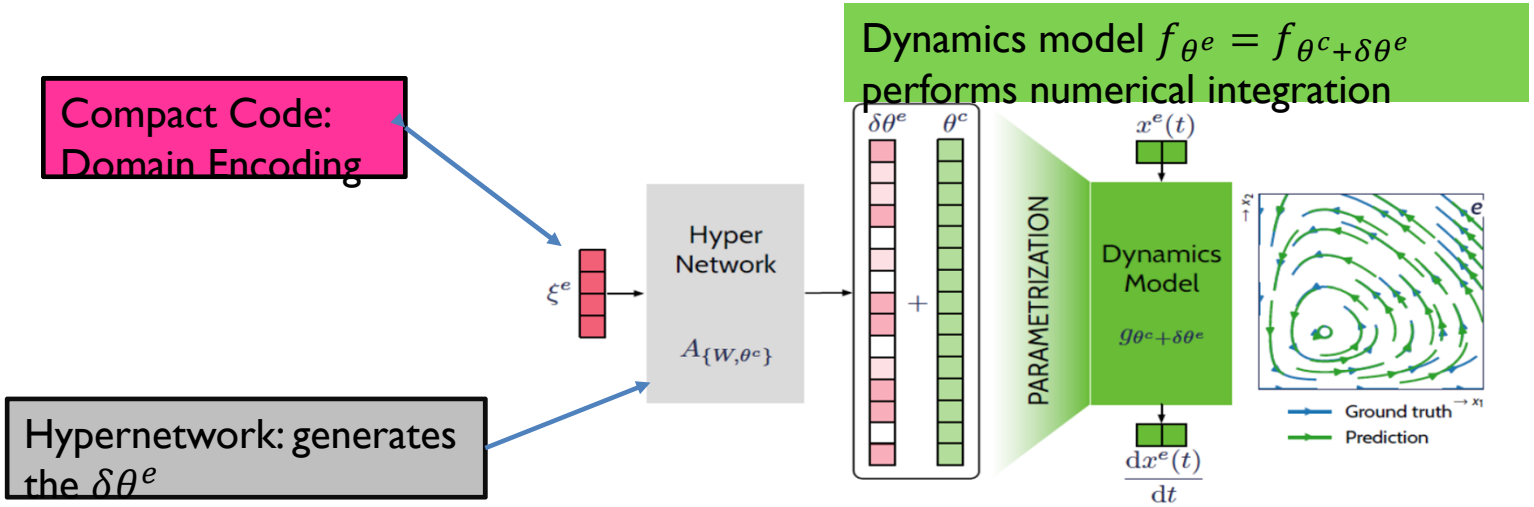
- Adaptation objective (inference)

$$\min_{\theta^c, W, \xi^e; e \in E} \sum_{e \in D_{Train}} \|W\xi^e\|^2 \text{ s.t. } \forall x^e \in D^e, \forall t, \frac{dx^e(t)}{dt} = f_{\theta^c + W\xi^e}\left(x^e(t)\right)$$

Adaptation is sample efficient and converges with few gradient steps
Reuses the knowledge from training domains

Tackling the generalization problem for dynamical systems
CODA framework (Kirchmeyer et al. 2022)

- Conditioning to an environment
  - Inference: for a new environment, shared parameters θ^c and W are fixed, learn code ξ^e in few shot and infer θ^e



Tackling the generalization problem for dynamical systems
CODA framework (Kirchmeyer et al. 2022)

- Dynamical function for environment e
  - f_{θ^e} with $\theta^e = \theta^c + \delta\theta^e$

- Lotka-Voltera ODE
  - Loss landscape for 3 environments
  - Centered on the shared $\boldsymbol{\theta^c}$
  - Local min θ^e indicated by **arrow**

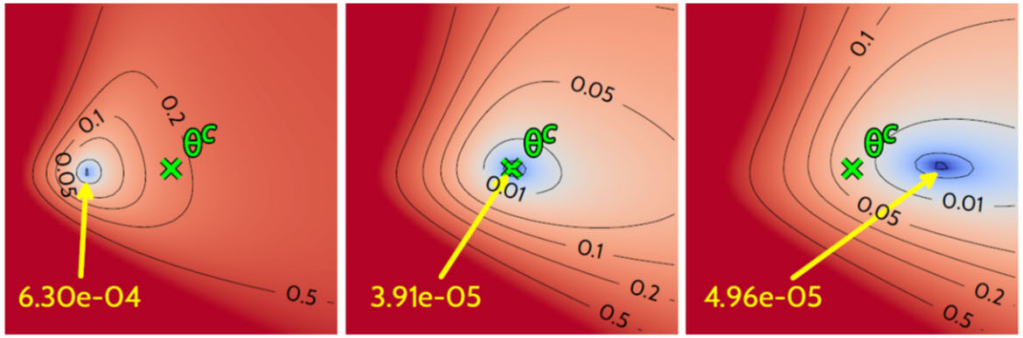


Figure 1. CoDA's loss landscape centered in θ^c, marked with ×, for 3 environments on the *Lotka-Volterra* ODE. Loss values are projected onto subspace $\mathcal{W}$, with $d_\xi = 2$. $\forall e$, $\rightarrow$ points to the local optimum $\theta^{e\star}$ with loss value reported in yellow.

- Effect of the locality contraint
  - Small $\delta\theta$: θ^e close to $\boldsymbol{\theta^c}$
  - Convex landscape
  - Fast adaptation

Tackling the generalization problem for dynamical systems
CODA framework (Kirchmeyer et al. 2022)

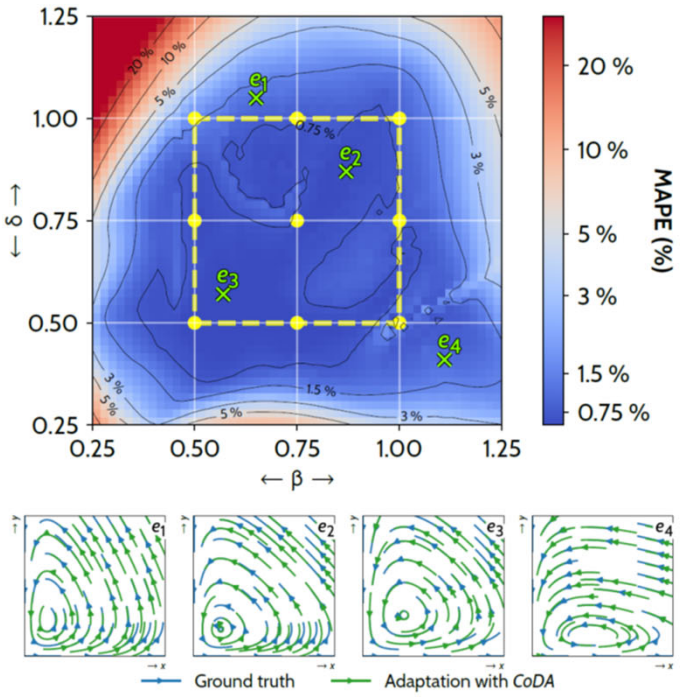


Figure 2. Adaptation results with CoDA-ℓ_1 on LV. Parameters (β, δ) are sampled in $[0.25, 1.25]^2$ on a 51×51 uniform grid, leading to 2601 adaptation environments $\mathcal{E}_{\text{ad}}$. • are training environments $\mathcal{E}_{\text{tr}}$. We report MAPE (↓) across $\mathcal{E}_{\text{ad}}$ (Top). On the bottom, we choose four of them (×, e_1–e_4), to show the ground-truth (blue) and predicted (green) phase space portraits. x, y are respectively the quantity of prey and predator in the system in Eq. (15).

Lotka-Volterra (LV, Lotka, 1925) The system describes the interaction between a prey-predator pair in an ecosystem, formalized into the following ODE:

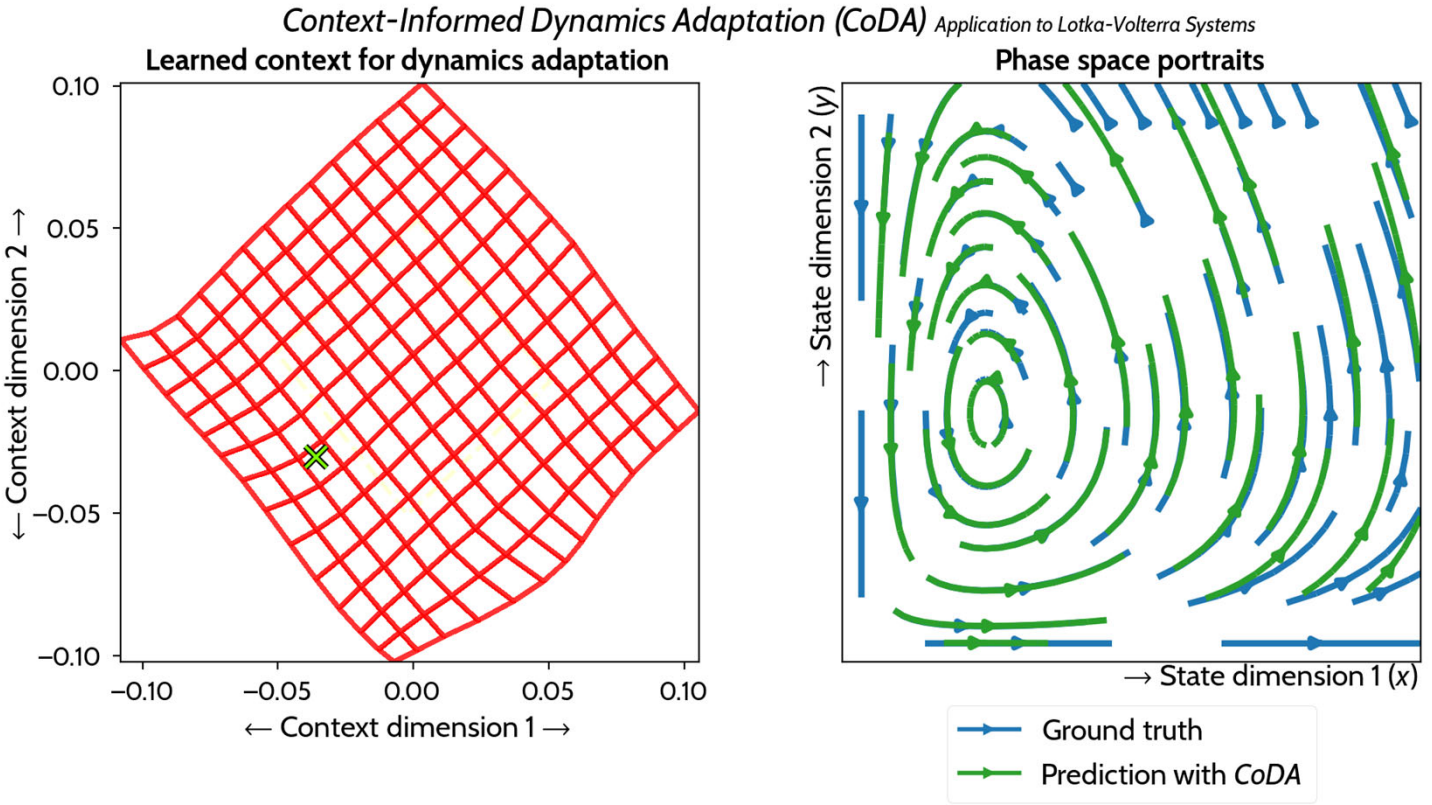
$$\begin{aligned} \frac{\mathrm{d}x}{\mathrm{d}t} &= \alpha x - \beta xy \\ \frac{\mathrm{d}y}{\mathrm{d}t} &= \delta xy - \gamma y \end{aligned} \tag{15}$$

where x, y are respectively the quantity of the prey and the predator, $\alpha, \beta, \delta, \gamma$ define how two species interact.

- Four parameters, two fixed (α, γ) and two (β, δ) change accross environments
- Training on 9 environments (yellow)
- Top: Evaluation on 2600 new environments
- Bottom: phase portraits for 4 new environments e_1 to e_4
  - **Blue trajectories: ground truth**
  - **Green trajectories: predicted**

Tackling the generalization problem for dynamical systems
CODA framework (Kirchmeyer et al. 2022)

- Visualization for Lotka-Voltera with a code ξ^e of size 2
- Learned code ξ^e is « isomorphic » to the 2 D parameter space



Tackling the generalization problem for dynamical systems
CODA framework (Kirchmeyer et al. 2022)

- Quantitative evaluation on extrapolation
  - Evaluation: MSE for 1-shot adaptation
  - Baselines: Multi-task and Meta-Learning

	Method	Lotka-Voltera (10^{-5})	Glycolic-Oscillator (10^{-4})	Gray-Scott (10^{-3})	Navier-Stokes (10^{-4})
Multi-task	LEADS	47.61	113.8	1.36	28.6
Meta-learning	MAML	3150	1081	2.25	48.6
Contextual meta-learning	CAVIA	6.26	2.37	1.62	26.0
	CODA	1.24	1.86	0.74	9.65

- Message
  - Dedicated strategy, adapted to the complexity of physics problems performs better than general purpose ML generalization/ adaptation

Tackling the generalization problem for dynamical systems: Generative approaches

ZEBRA: In-context generative pretraining for solving parametric PDEs (Serrano et al. ICML 2025)

ZEBRA - In-context generative pretraining (Serrano et al. 2025)

- Inspired by In-context learning in NLP decoders (LLMs)

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
1  Translate English to French:        ← task description
2  sea otter => loutre de mer          ← examples
3  peppermint => menthe poivrée        ←
4  plush girafe => girafe peluche      ←
5  cheese =>                           ← prompt
```

Fig. Brown et al. 2020 (GPT3)

```
Input: 2014-06-01
Output: !06!01!2014!
Input: 2007-12-13
Output: !12!13!2007!
Input: 2010-09-23
Output: !09!23!2010!
Input: 2005-07-23
Output: !07!23!2005!
```

in-context examples

test example

model completion

Fig. https://ai.stanford.edu/blog:...

No gradient update – only context

ZEBRA - In-context generative pretraining (Serrano et al. 2025)

How does in-context learning works: Two main interpretations

Gradient update interpretation

Dai et al. ACL 2023. Why can, GPT learn in context ?
Transformer attention has a dual interpretation as gradient descent in the linear attention case.

Interpret LLM as meta-optimizers that perform implicit fine tuning for in-context examples.

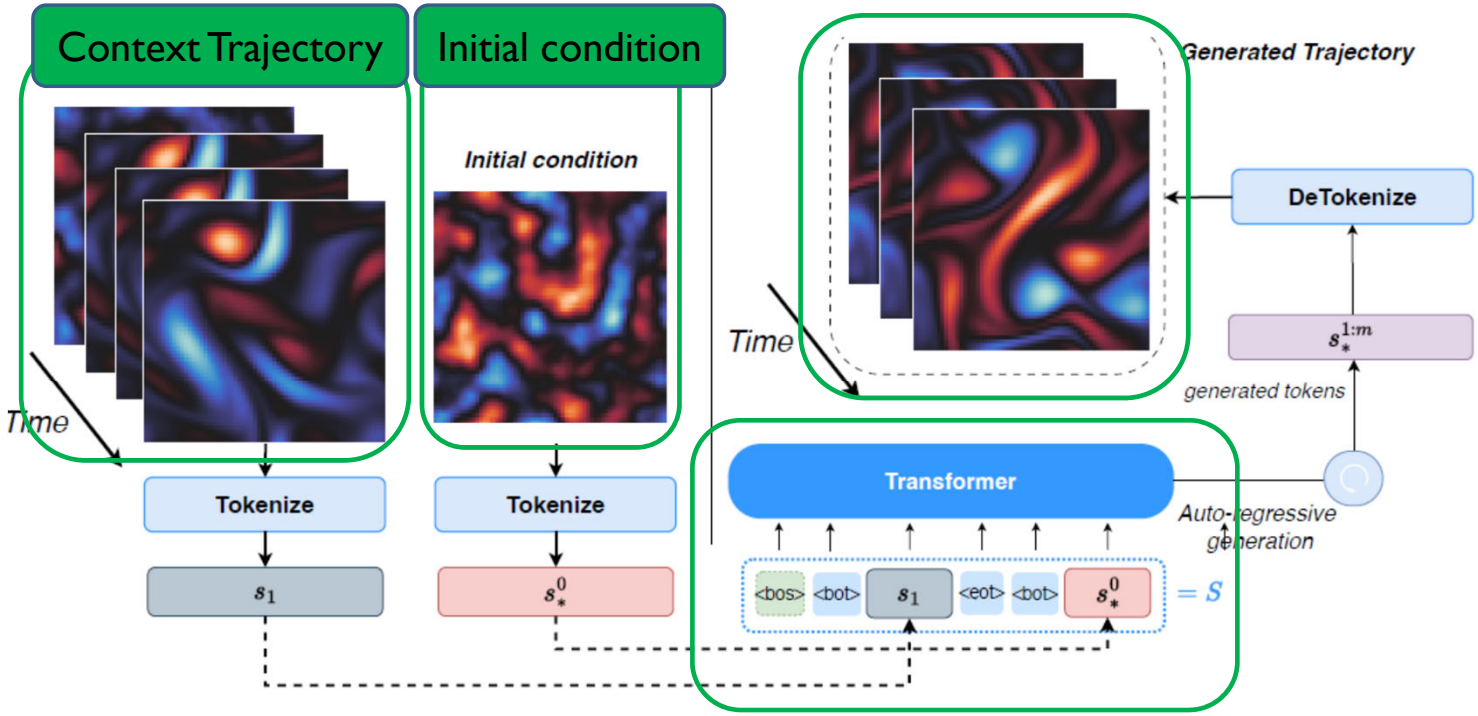
Bayesian interpretation

Xie et al. https://ai.stanford.edu/blog/understanding-incontext/

Pretraining learns latent concept distributions, inference identifies the promp latent concept

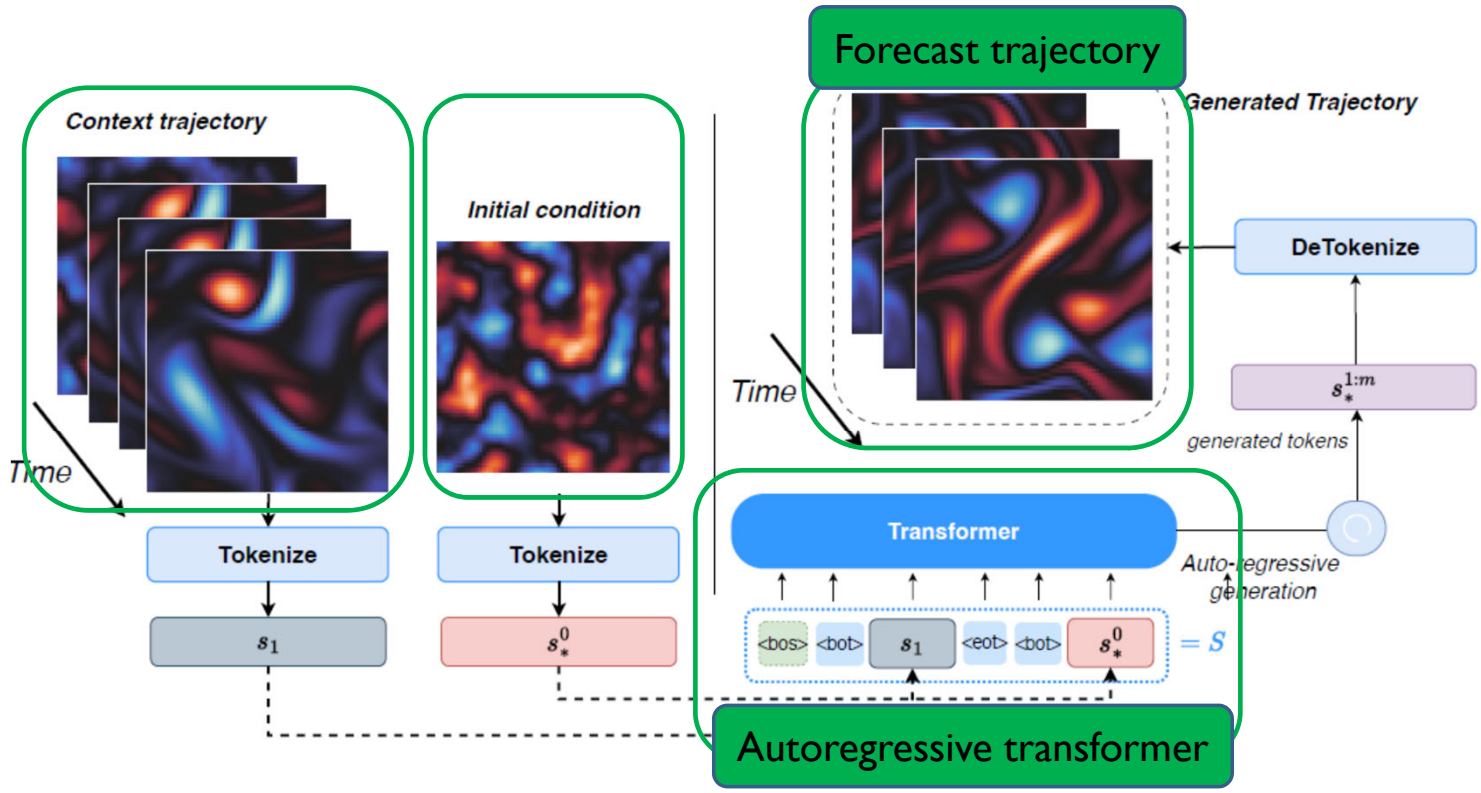
$$p(\text{output}|\text{prompt}) = \int_{\text{concept}} p(\text{output}|\text{concept}, \text{prompt}) p(\text{concept}|\text{prompt}) d(\text{concept})$$

ZEBRA - In-context generative pretraining (Serrano et al. 2025)
Inference

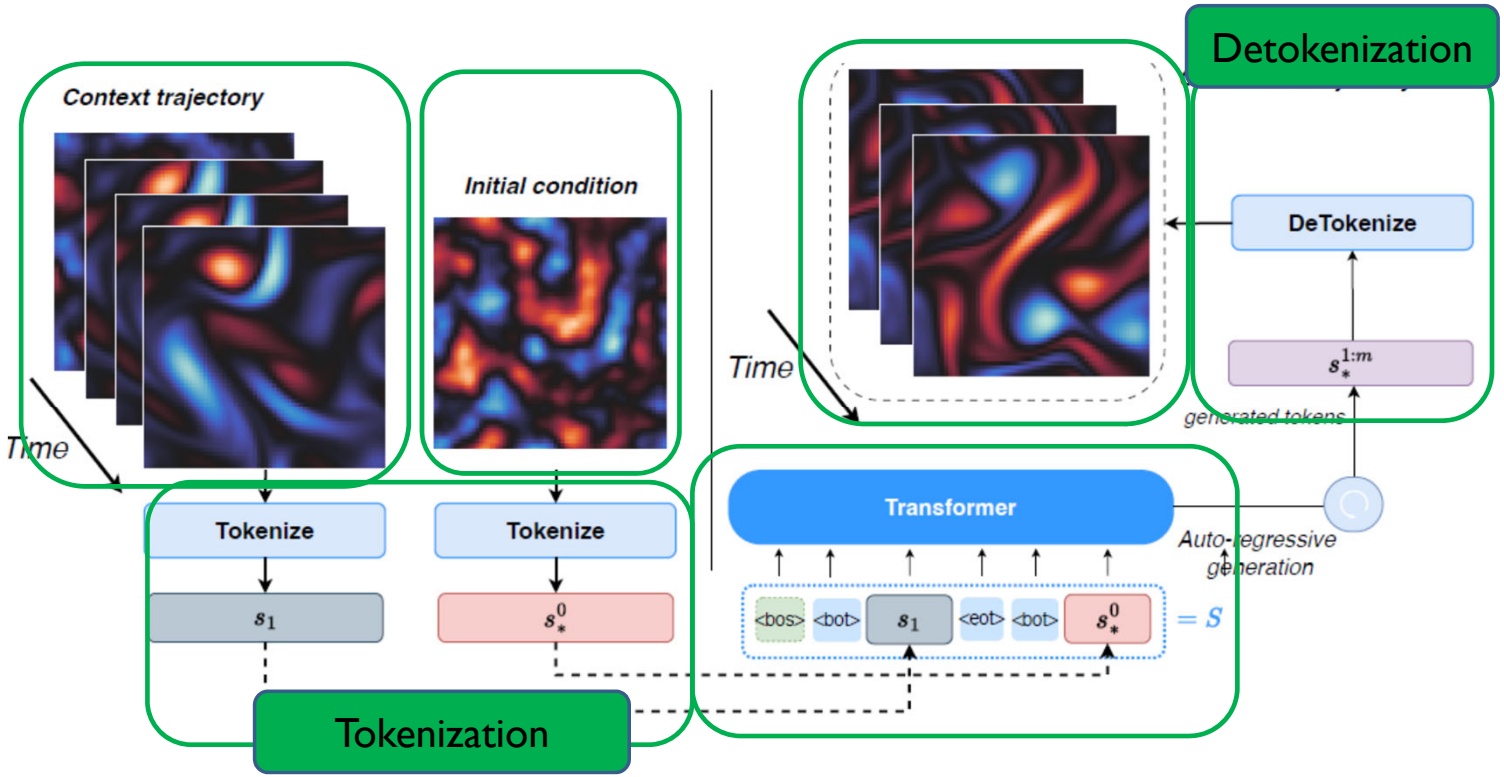


Context trajectory: trajectory from the same PDEs starting from another Initial Condition

ZEBRA - In-context generative pretraining (Serrano et al. 2025)
Inference



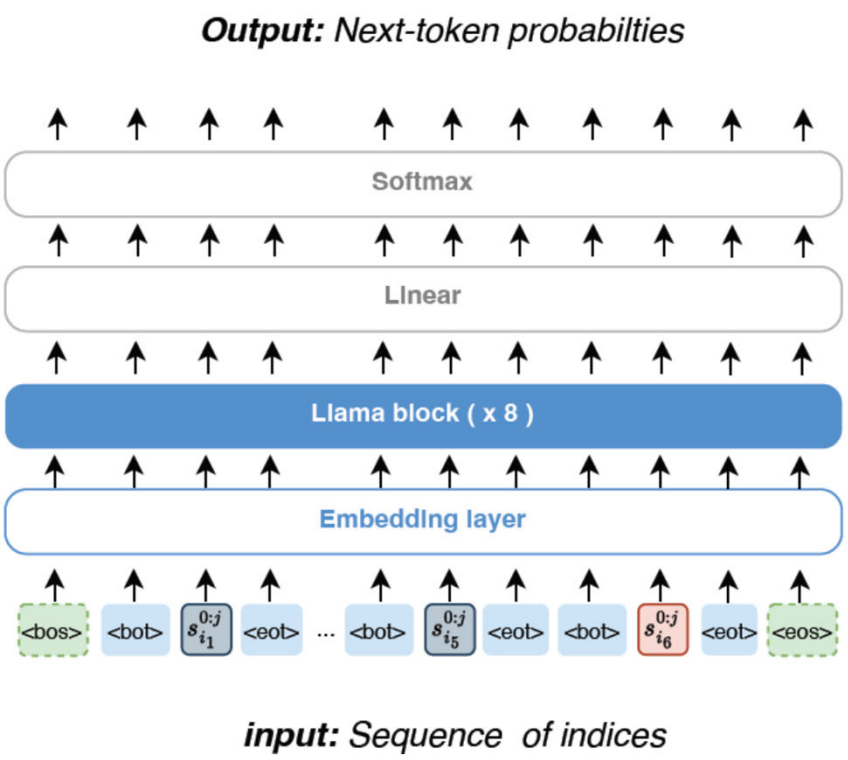
ZEBRA - In-context generative pretraining (Serrano et al. 2025)
Inference: **discrete model** i.e. LLM structure

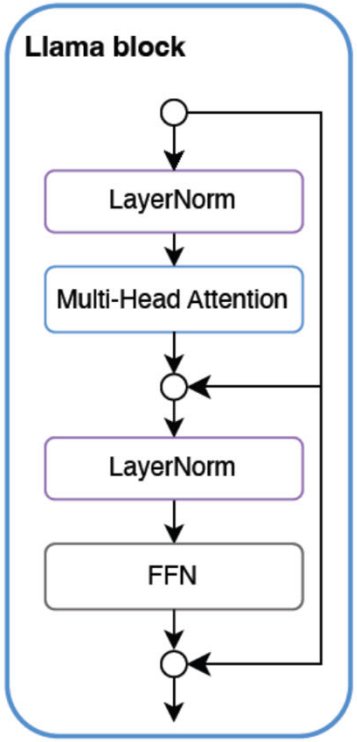


The transformer operates on discrete tokenized representations of the data

ZEBRA - In-context generative pretraining (Serrano et al. 2025)

- Transformer is a Llama like architecture



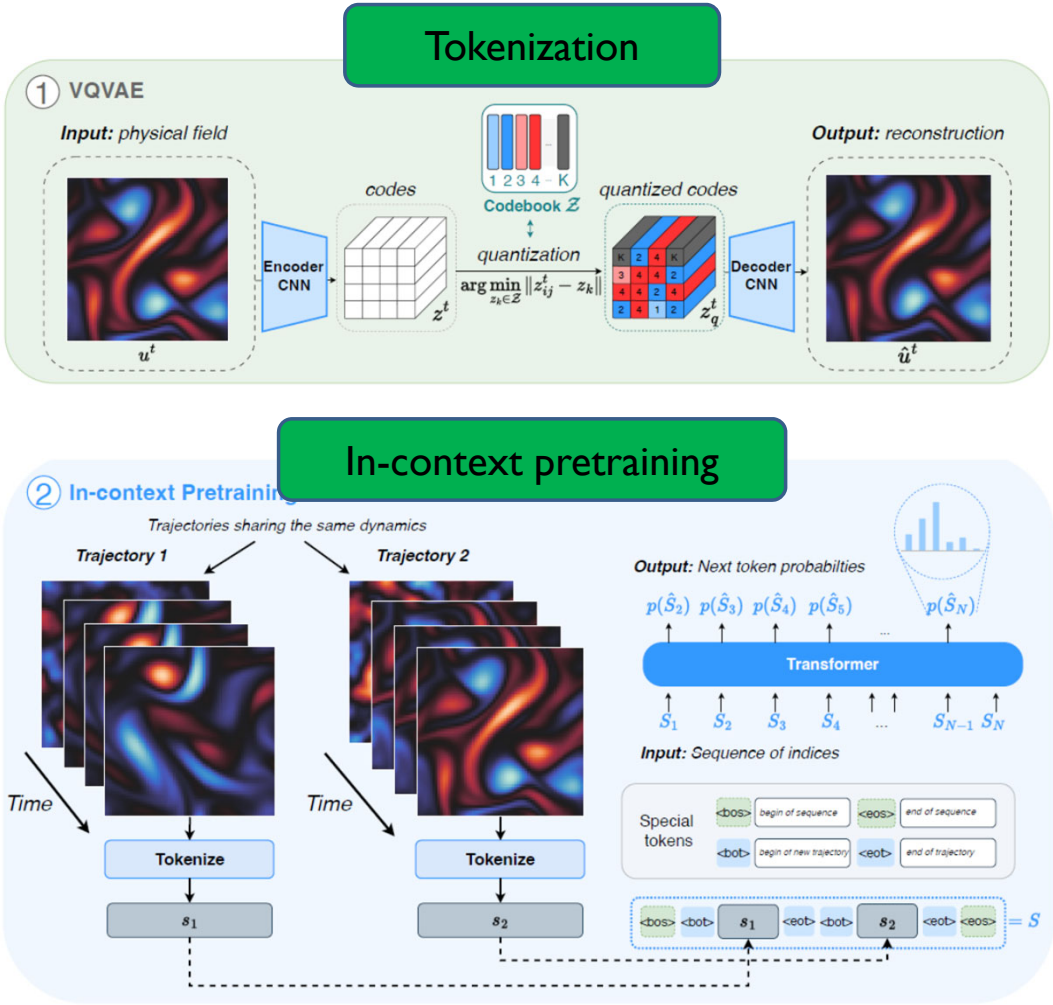


ZEBRA learns trajectory distributions

ZEBRA - In-context generative pretraining (Serrano et al. 2025)
Training: 2 steps

Encode-decode:VQVAE
Maps a frame to a finite set of tokens

Processor: LLM
Predicts discrete distribution of **tokens**
Training loss: cross entropy



ZEBRA - In-context generative pretraining (Serrano et al. 2025)

Examples

Heat equation

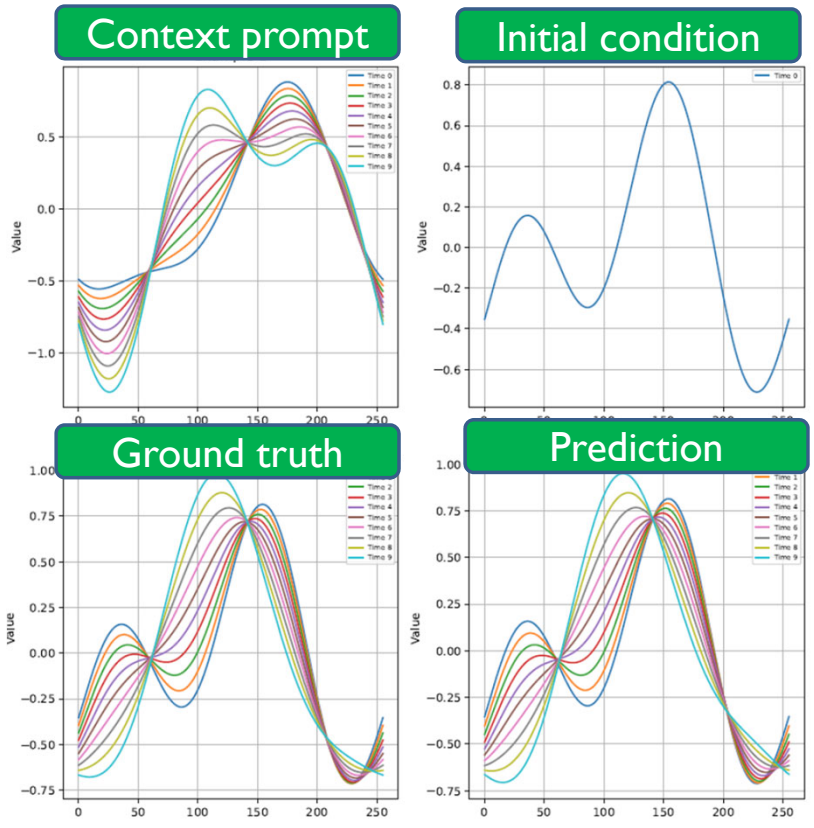


Figure 26: **One-shot** adaptation on Heat

Combined equation

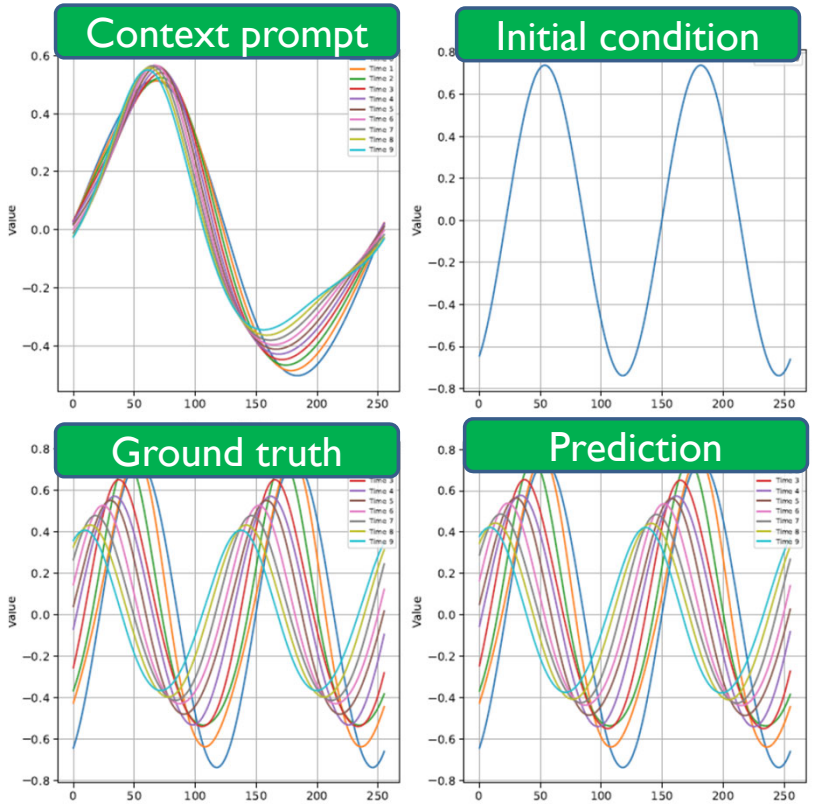


Figure 30: **One-shot** adaptation on Combined

ZEBRA - In-context generative pretraining (Serrano et al. 2025)
Examples

Vorticity

Context prompt

Ground truth

Perdiction

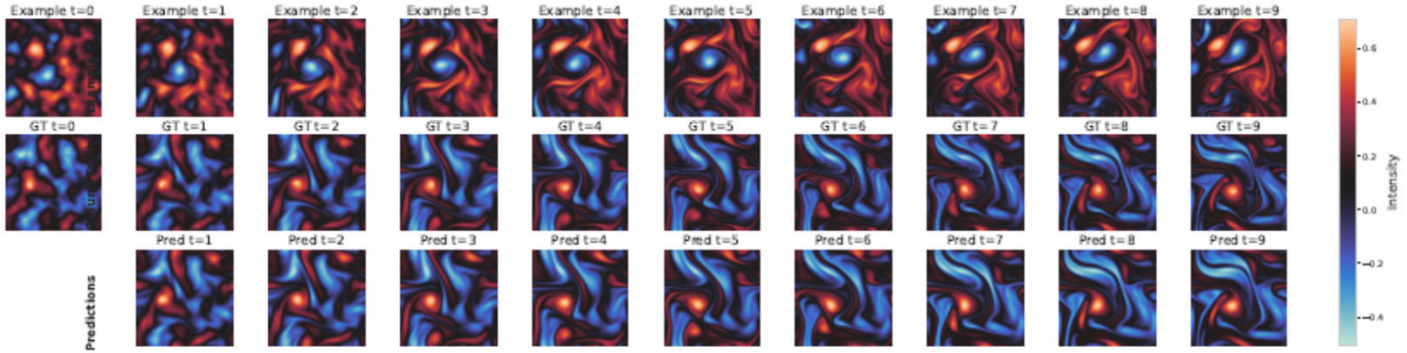


Figure 35: **One-shot OoD** adaptation on Vorticity. Example 1.

ZEBRA - In-context generative pretraining (Serrano et al. 2025) evaluation

In-distribution- one shot adaptation

Table 1: One-shot adaptation. Conditioning from a similar trajectory. Test results in relative L2 on the trajectory. '–' indicates inference has diverged.

Gradient based adaptation

Visual transformers

	Advection	*Heat*	*Burgers*	*Wave b*	*Combined*	*Wave 2D*	*Vorticity 2D*
CAPE	0.00941	0.223	0.213	0.978	**0.00857**	–	–
CODA	**0.00687**	0.546	0.767	1.020	0.0120	0.777	0.678
[CLS] ViT	0.140	**0.136**	0.116	0.971	0.0446	0.271	0.972
ViT-in-context	0.0902	0.472	0.582	0.472	0.0885	0.390	0.173
Zebra	0.00794	0.154	**0.115**	**0.245**	0.00965	**0.207**	**0.119**

Out of-distribution- one shot adaptation

Table 2: Out-of-distribution results. Test results in relative L2 on the trajectory. '–' indicates inference has diverged.

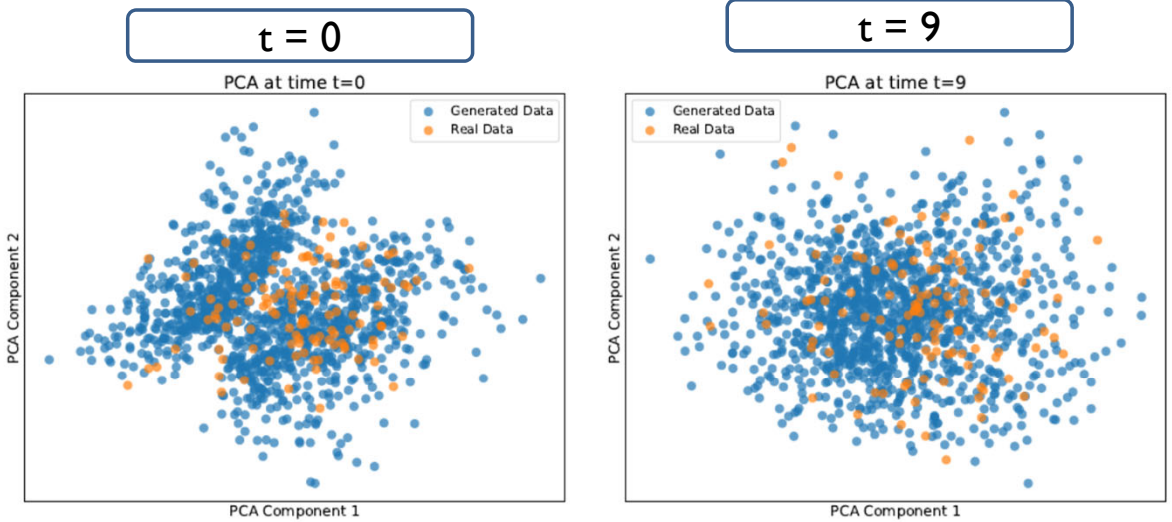
	Heat	*Wave 2D*	*Vorticity 2D*	
			close	*far*
CAPE	0.47	–	–	–
CODA	1.03	1.51	1.71	–
ViT-in-context	0.52	**0.68**	0.30	0.368
Zebra	**0.15**	**0.68**	**0.24**	**0.317**

ZEBRA - In-context generative pretraining (Serrano et al. 2025)
Trajectory generation

- Trajectory generation: prompting with an in-context example, generate a trajectory without giving an initial condition
- The distributions of the generated and actual data are similar (combined equation)

Generated distribution

Target distribution



(a) Distribution of generated initial conditions ($t = 0$). (b) Distribution of generated trajectories at ($t = 9$).

Figure 19: Qualitative analysis of generated trajectories. `Zebra` generates new initial conditions and trajectories for unseen test environments. PCA projections visualize both generated and true trajectories in a lower-dimensional space at $t = 0$ and $t = 9$.

ZEBRA - In-context generative pretraining (Serrano et al. 2025)
Trajectory generation

- Trajectory generation: prompting with an in-context example, generate a trajectory without giving an initial condition
- Example Vorticity 2D

Prompt

Generated trajectory

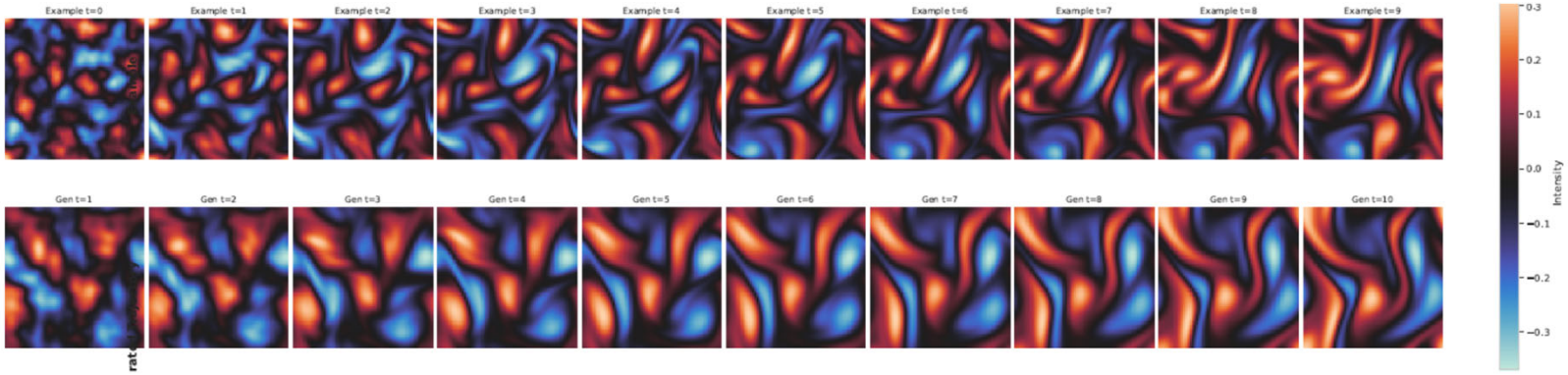


Figure 18: **Unconditional generation** on *Vorticity 2D*. The top-row is the example used to guide the generation, and the bottom-row is the generated example. The model also generates the initial condition.

ZEBRA - In-context generative pretraining (Serrano et al. 2025)
Uncertainty quantification

- Example: Heat equation
- Generate multiple trajectories: mean trajectory and confidence interval +/-3 standard deviation

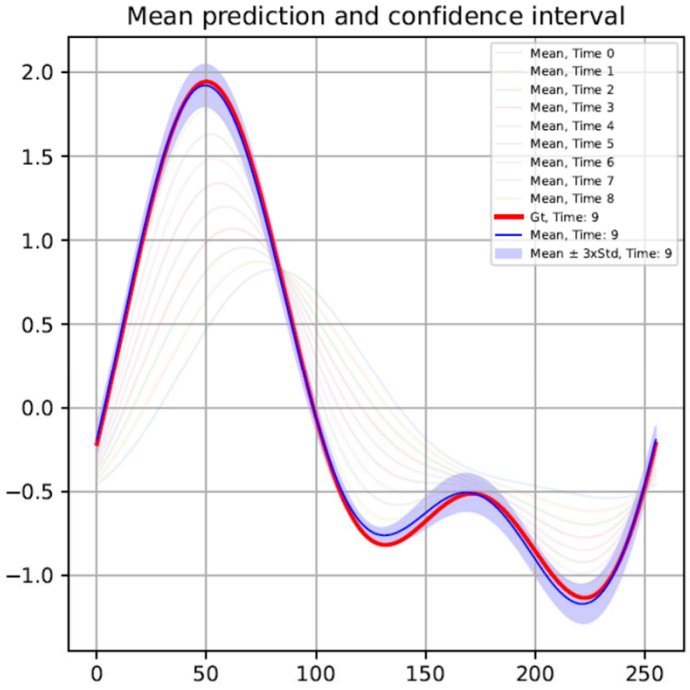


Figure 16: **Uncertainty quantification** with `Zebra` in a one-shot setting on *Heat* equation

Quantitative evaluation: CRPS and RMSCE

- CRPS and RMSCE Results

Metric	Model	Advection	Heat	Burgers	Wave b	Combined
CRPS	ViT + noise	0.0705	0.176	0.227	0.093	0.098
	ViT Dropout	0.0363	0.213	0.196	0.024	0.024
	Zebra	**0.0026**	**0.043**	**0.020**	**0.0129**	**0.0018**
RMSCE	ViT + noise	0.132	0.241	0.265	0.249	**0.045**
	ViT Dropout	0.386	0.547	0.529	0.340	0.064
	Zebra	**0.074**	**0.055**	**0.048**	**0.124**	0.074

CRPS: Continuous Ranked probability Score: accuracy of forecast
RMSCE: Root Mean Squared Calibration Error: calibration of the forecast

Tackling the generalization problem for dynamical systems: Generative approaches

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Two main families of generative models are competing for large scale applications for vision and NLP

Auto-regressive discrete models leveraging LLMs

- Initially developed for NLP (e.g. GPT decoders) and more recently adapted to vision - Leverage discrete tokens

Continuous one shot models, e.g. diffusion models

- Initially developed for image and video generation (e.g. Stability AI's Stable Diffusion, OpenAI DALL-E, Google Imagen)

Comparison for video generation

Advantage: Scalable to longer videos, less GPU memory required per step.
Disadvantage: Risk of temporal drift, flickering, or inconsistency over time

Advantage: Strong temporal coherence across frames. No error accumulation over time.
Disadvantage: Memory- and compute-intensive, especially for long or high-res videos

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Two main families of generative mod… for large scale appli…

Auto-regressi…

- Initial… decod… vision -…

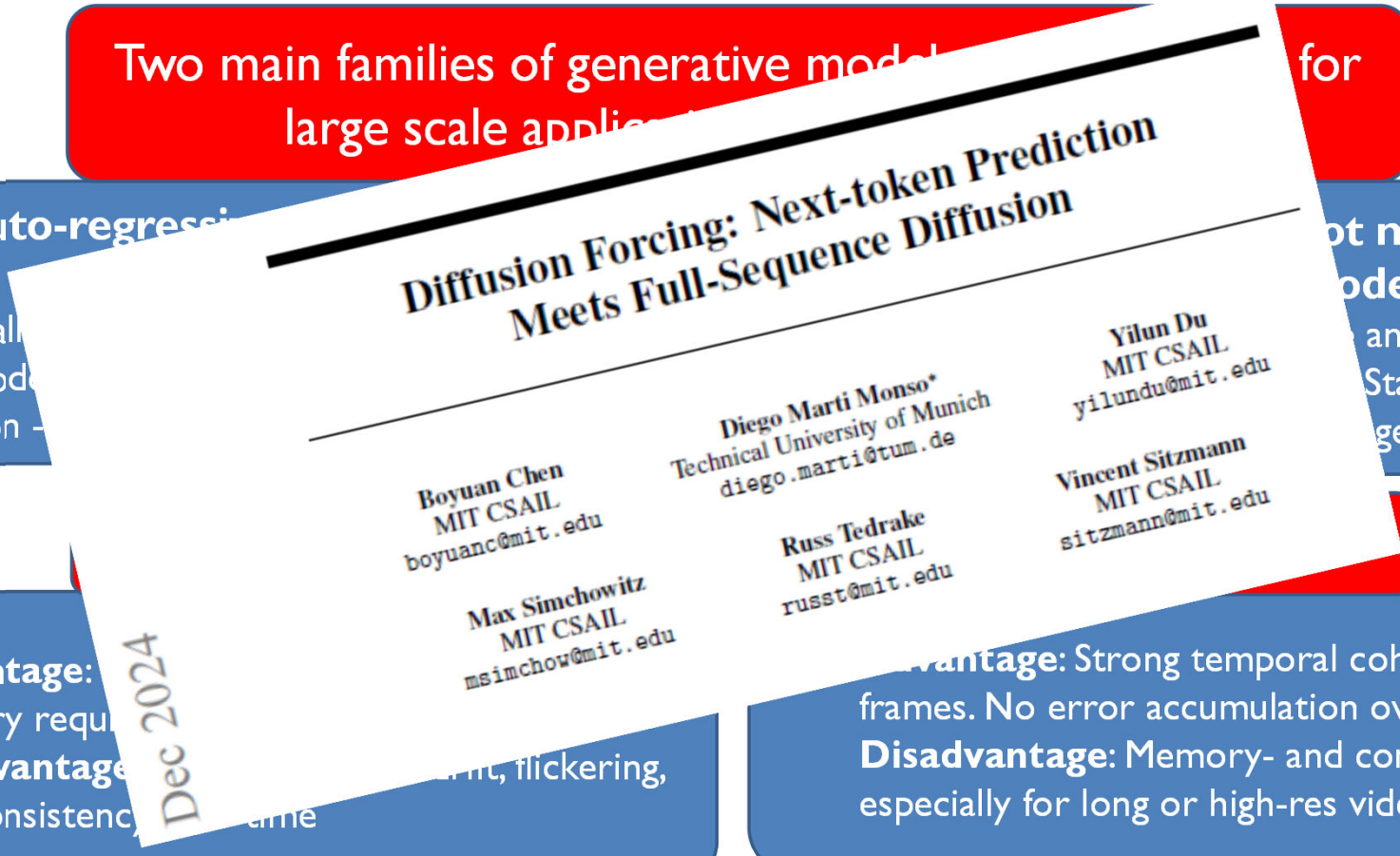
…ot models, e.g. …odels

…and video …Stable Diffusion, …gen)

Advantage: … memory requ… **Disadvantage:** … or inconsistenc… …ime

…ft, flickering,

…vantage: Strong temporal coherence across frames. No error accumulation over time.
Disadvantage: Memory- and compute-intensive, especially for long or high-res videos



Diffusion Forcing: Next-token Prediction Meets Full-Sequence Diffusion

Boyuan Chen
MIT CSAIL
boyuanc@mit.edu

Diego Marti Monso*
Technical University of Munich
diego.marti@tum.de

Yilun Du
MIT CSAIL
yilundu@mit.edu

Max Simchowitz
MIT CSAIL
msimchow@mit.edu

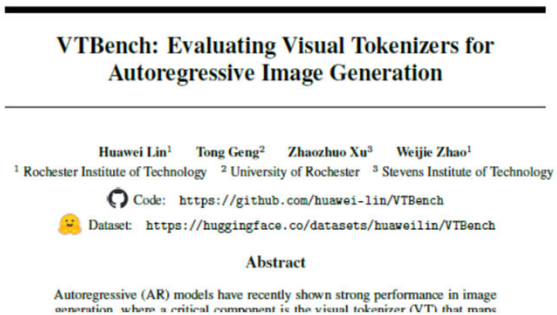
Russ Tedrake
MIT CSAIL
russt@mit.edu

Vincent Sitzmann
MIT CSAIL
sitzmann@mit.edu

Dec 2024

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

- Autoregressive generation is a natural approach for temporal/ spatio-temporal modeling
- Current generative autoregressive models leverage discrete LLM models for predicting discrete probabilities
  - Loss of information due to discrete tokens encoding



May 2025

VTBench: Evaluating Visual Tokenizers for Autoregressive Image Generation

Huawei Lin[1] Tong Geng[2] Zhaozhuo Xu[3] Weijie Zhao[1]
[1] Rochester Institute of Technology [2] University of Rochester [3] Stevens Institute of Technology

Code: https://github.com/huawei-lin/VTBench
Dataset: https://huggingface.co/datasets/huaweilin/VTBench

Abstract

Autoregressive (AR) models have recently shown strong performance in image generation, where a critical component is the visual tokenizer (VT) that maps continuous pixel inputs to discrete token sequences. The quality of the VT largely defines the upper bound of AR model performance. However, current discrete VTs fall significantly behind continuous variational autoencoders (VAEs), leading to degraded image reconstructions and poor preservation of details and text. Existing benchmarks focus on end-to-end generation quality, without isolating VT perfor-
mance. To address this gap, we introduce VTBench, a comprehensive benchmark

- Operating in a continuous space appears more natural for modeling physical dynamics
- ENMA explores autoregressive models operating in a continuous space

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Flow matching

Generative approach that

- Learns a probability path $p_t(z)$ from a source distribution $p_0(z)$ to a target one $p_1(z)$
- $p_t(z)$ is modeled with continuous-time velocity field $v_\theta(z,t)$, describing the instantaneous velocity of samples $z(t)$
- $v_\theta(z,t)$ is trained to approximate a target velocity field defined by an ODE $\frac{dz(t)}{dt} = v^*(z,t)$

Example with a linear interpolation trajectory

$$z(t) = (1-t)\epsilon + tz, \quad \epsilon \sim p_0(z) = \mathcal{N}(O,I), z \sim p_1(z)$$

$$v^*(z,t) = z - \epsilon$$

Training loss

$$L(\theta) = E_{\epsilon,\mathrm{z,t}}[\|v_\theta(z(t),t) - v(z,t)\|^2] = E_{\epsilon,\mathrm{z,t}}[\|v_\theta(z(t),t) - z - \epsilon\|^2]$$

Inference

Sample $\epsilon \sim \mathcal{N}(O,I)$

Solve the IVP $\frac{dz}{dt} = v_\theta(z(t),t), \; z(0) = \epsilon$ to obtain a sample from the target distribution p_1

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Discrete vs continuous token distribution forecasting

- Consider a frame with M tokens $Z = (z_1, \dots, z_M)$

Next token prediction $p(z_1, \dots, z_M) = \sum_{i=1}^{M} p(z_i | z_{<i})$

How $p(z_i | z_{<i})$ is modeled:

- Compute a context $c_i = f(z_1, \dots, z_i)$, e.g. with a transformer
- Model next token probability $p(z_i | c_i)$

Discrete case

$z_i \in \{1, \dots, K\}$

- $p(z_i = k | c_i) = softmax\ (logit(c_i, k))$
- $logit(c_i, k)$ is the logit corresponding to class k

Continuous case

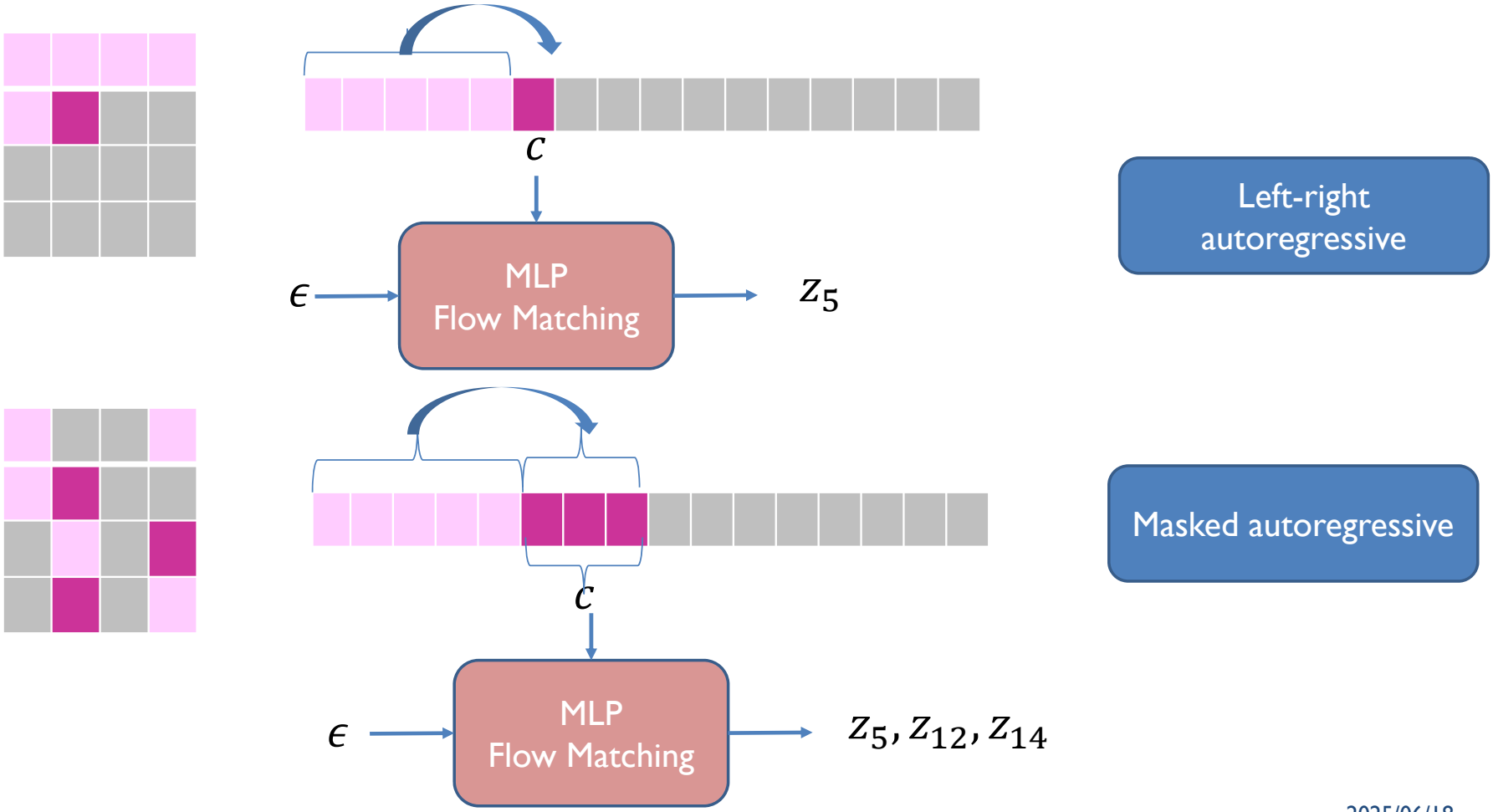
$z_i \in R^d$

Sample $\epsilon \sim \mathcal{N}(O, I)$

Solve $\frac{dz(t)}{dt} = v_\theta(z(t), t, c_i)$, $z(0) = \epsilon$ to get a sample z_i

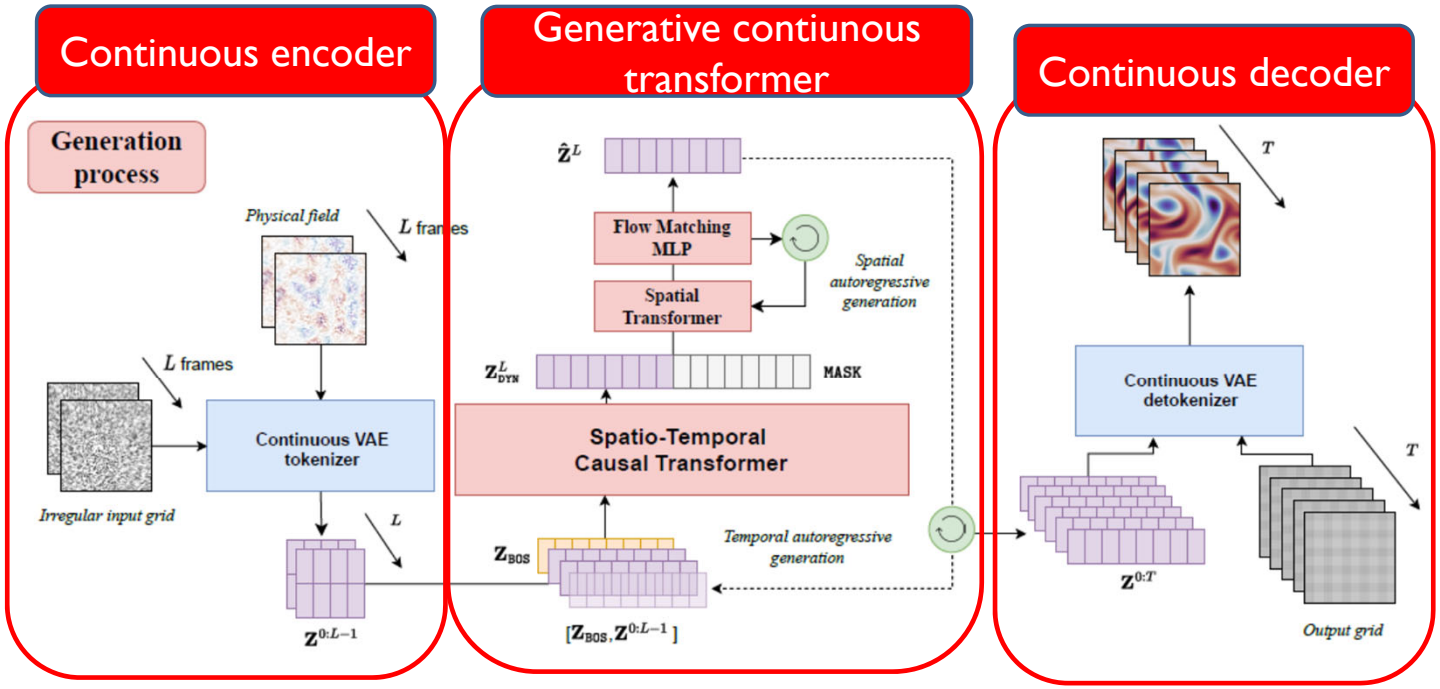
ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

From Autoregression to to masked autoregression (Li et al. 2024 - https://arxiv.org/abs/2406.11838)



ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

General structure

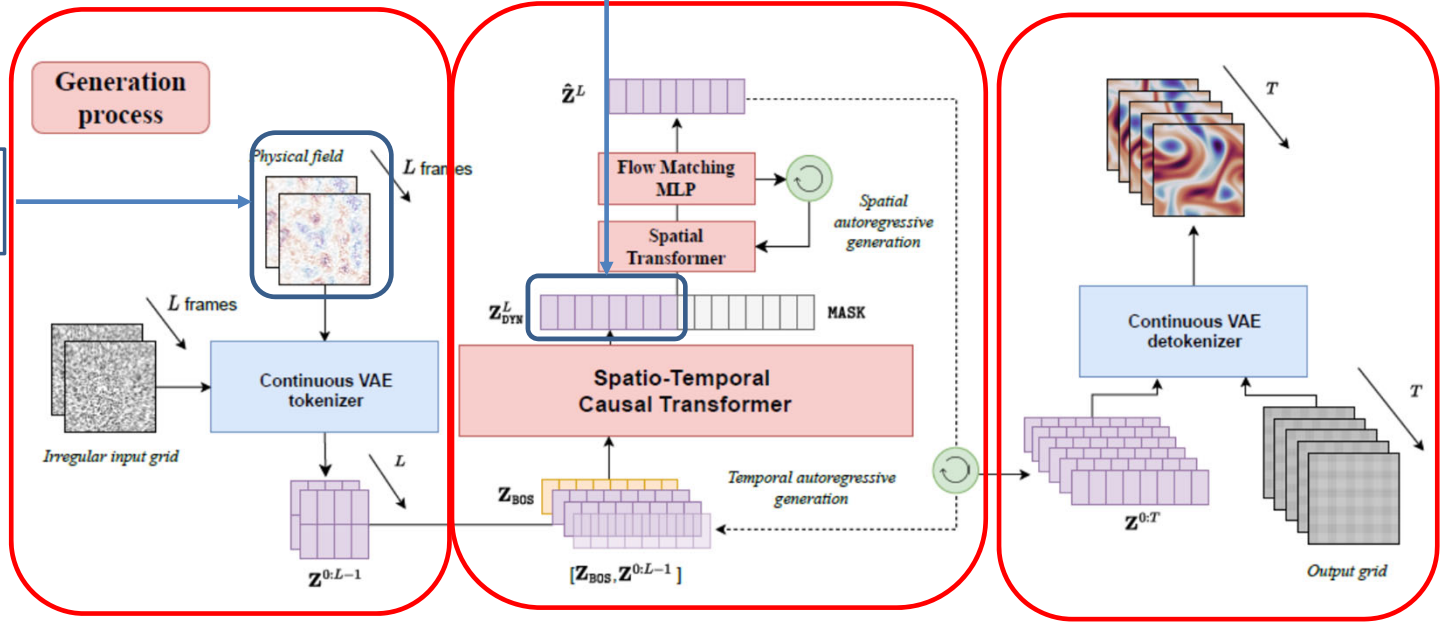


ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

General structure

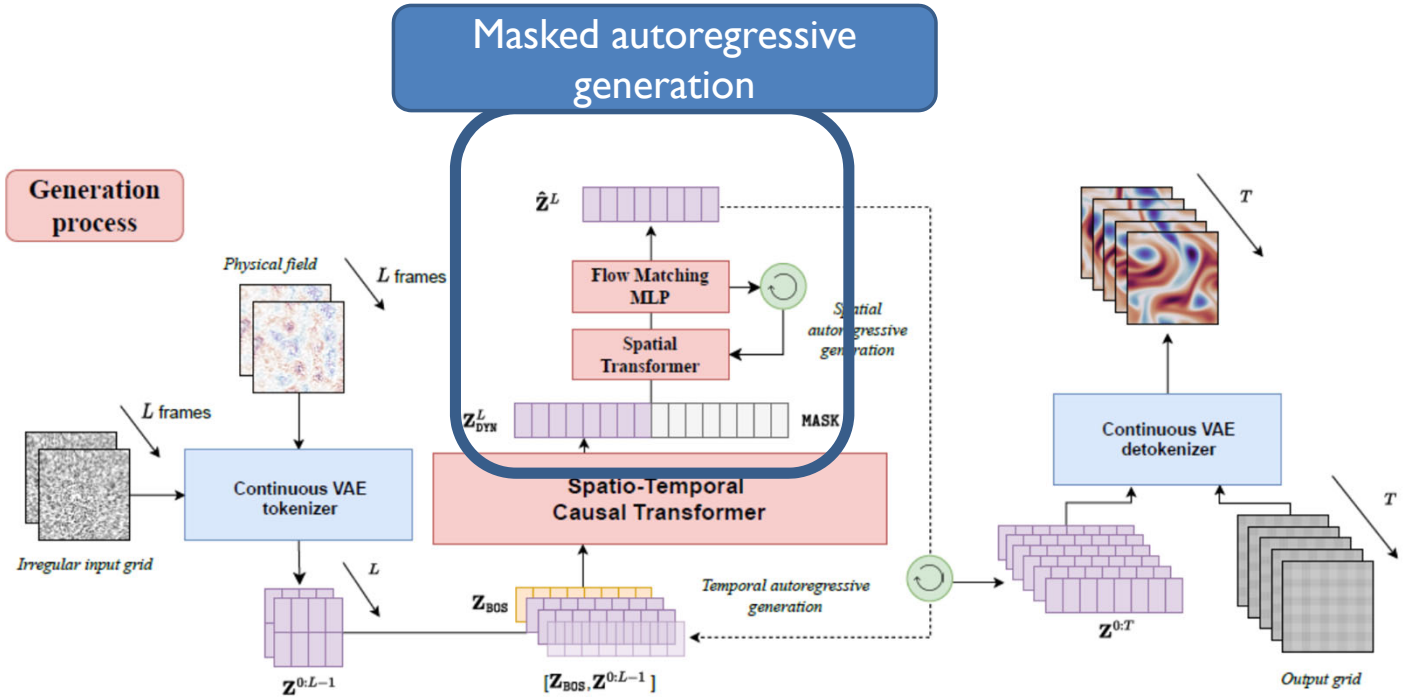
Fixed size latent representation of the spatio temporal context

Historical context L frames



ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Generative component

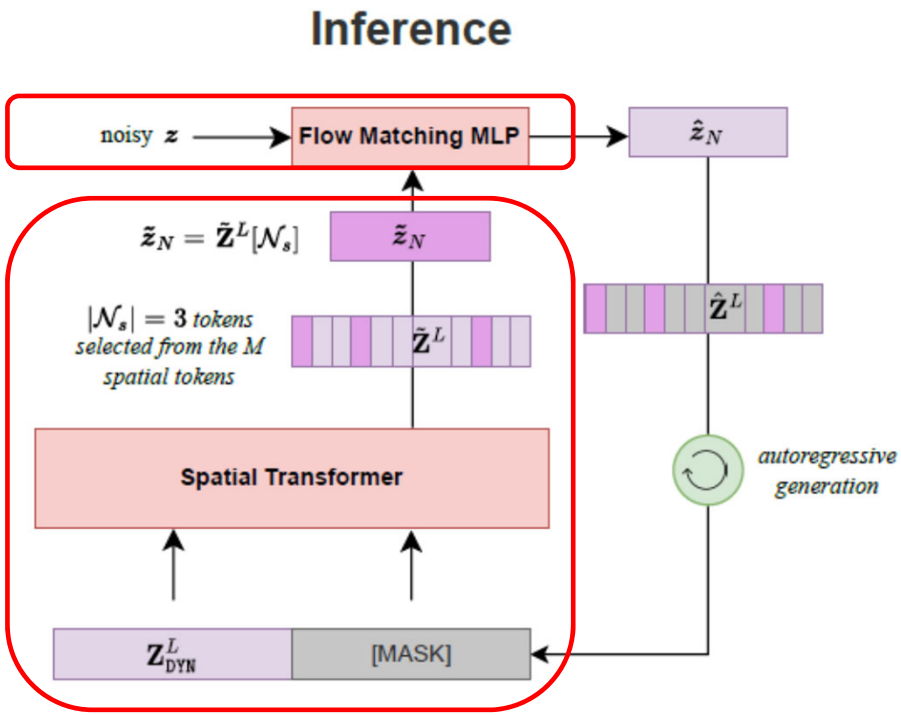


ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Generative component

Stochastic component
Flow matching

Deterministic component
Masked AR spatial transformer



ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Generative component - iterations

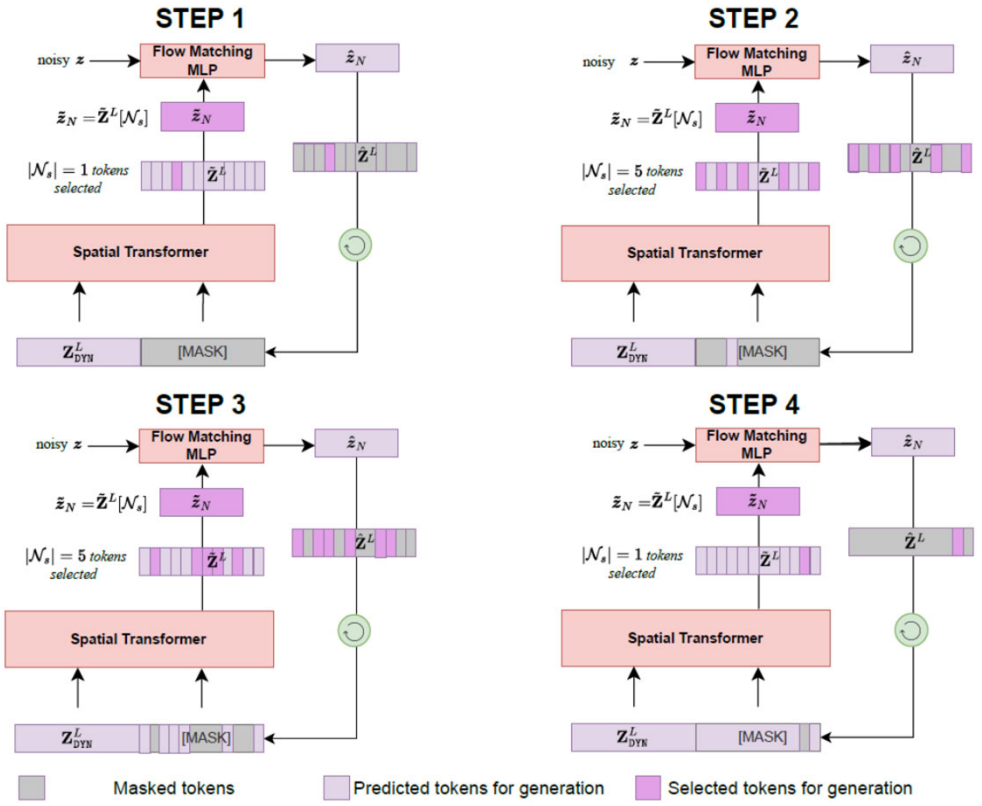


Figure 19: Illustration of tokenwise autoregressive generation at inference with a spatial Transformer over $S = 4$ steps. At each step, a subset of tokens is selected for generation based on a cosine schedule.

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)
Evaluation

- In and out distribution performance

Table 2: Comparison of model performance for temporal conditioning and initial value problem tasks across 5 dynamical systems. Metrics in Relative MSE. Lower is better.

Setting ↓	Dataset →	Advection		Combined		Gray-Scott		Wave		Vorticity	
	Model ↓	*In-D*	*Out-D*	*In-D*	*Out-D*	*In-D*	*Out-D*	*In-D*	*Out-D*	*In-D*	*Out-D*
Temporal Conditioning	FNO	2.47e-1	7.95e-1	1.33e-1	2.66e+1	5.04e-2	1.92e-1	6.91e-1	2.64e+0	6.07e-2	**2.15e-1**
	BCAT	5.55e-1	9.23e-1	2.68e-1	9.28e-1	3.74e-2	1.57e-1	2.19e-1	5.38e-1	5.39e-2	3.00e-1
	AVIT	1.64e-1	**5.02e-1**	5.67e-2	3.05e-1	4.26e-2	1.68e-1	1.57e-1	5.88e-1	1.76e-1	3.77e-1
	AR-DiT	2.36e-1	8.56e-1	2.95e-1	1.80e+0	3.69e-1	4.99e-1	1.12e+0	7.52e+0	1.98e-1	4.80e-1
	Zebra	2.04e-1	1.39e+0	1.82e-2	2.20e+0	4.21e-2	1.82e-1	**1.40e-1**	**3.15e-1**	**4.43e-2**	2.23e-1
	ENMA	**3.95e-2**	5.30e-1	**7.86e-3**	**1.02e-1**	**3.40e-2**	**1.44e-1**	1.45e-1	4.89e-1	7.58e-2	3.45e-1
Initial Value Problem	In-Context ViT	1.15e+0	1.20e+0	5.79e-1	1.36e+0	6.90e-2	1.94e-1	1.72e-1	6.24e-1	1.53e-1	3.92e-1
	[CLS] ViT	1.15e+0	1.36e+0	9.60e-2	1.16e+0	4.80e-2	2.19e-1	5.56e-1	1.02e+0	**4.30e-2**	2.59e-1
	Zebra	3.16e-1	1.47e+0	4.78e-2	9.63e-1	**4.40e-2**	**1.22e-1**	1.69e-1	**3.52e-1**	5.90e-2	**2.29e-1**
	ENMA	**2.02e-1**	**8.07e-1**	**1.56e-2**	**3.30e-1**	4.80e-2	1.34e-1	**1.54e-1**	5.02e-1	8.58e-2	3.20e-1

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Evaluation

ENMA requires only a small number of autoregressive steps (Left) and flow matching steps (right)

(Combined equation)

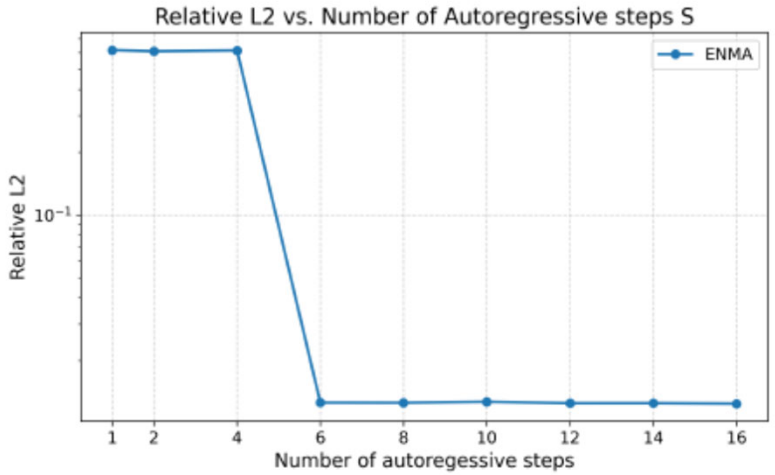


Figure 25: Relative L2 error as a function of the number of autoregressive steps S. Performance improves markedly around $S = 6$ and plateaus thereafter.

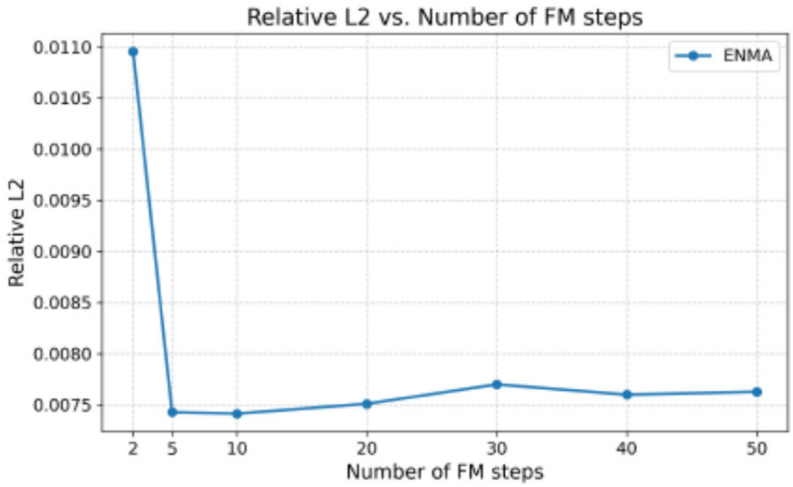


Figure 26: Relative L2 error versus number of flow matching steps per token. Performance quickly stabilizes after 5 steps.

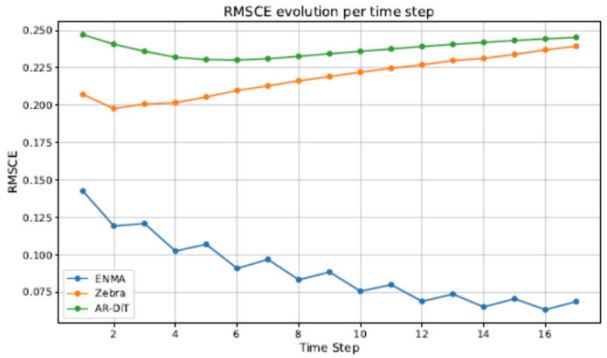
ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Uncertainty quantification

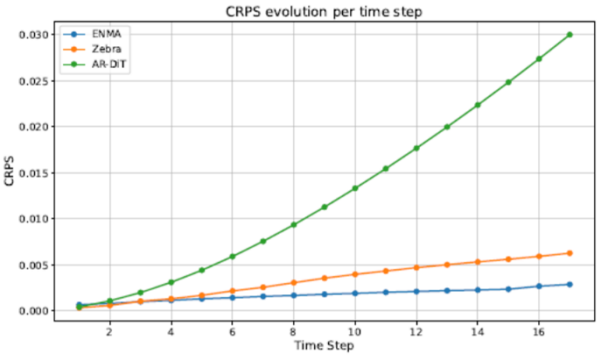
- Generate multiple trajectories and compute statistics

Method	Metric	Combined
AR-DiT	RMSCE	2.68e-1
	CRPS	1.27e-2
Zebra	RMSCE	2.19e-1
	CRPS	9.00e-3
ENMA (ours)	RMSCE	**8.68e-2**
	CRPS	**1.70e-3**

Table 12: Comparison of uncertainty metrics (↓ is better) for Combined.



(a) RMSCE evolution over time. Lower values indicate better uncertainty calibration.



(b) CRPS evolution over time. Lower values reflect sharper and more accurate forecasts.

RMSCE over time

CRPS over time

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Examples: Gray Scott

In-distribution

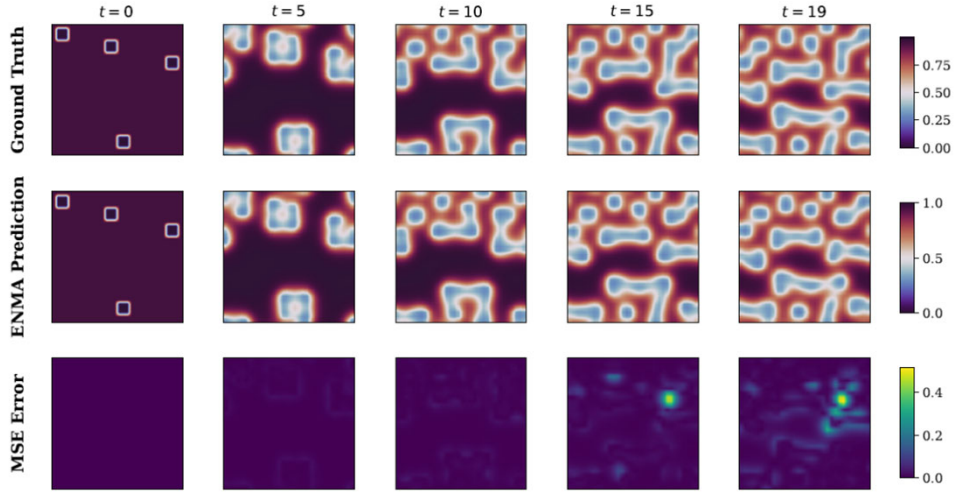


Figure 37: Qualitative comparison between ENMA prediction and ground truth for an in-distribution sample from the Gray-Scott dataset ($F = 0.0323$, $k = 0.0606$).

Out-of-distribution

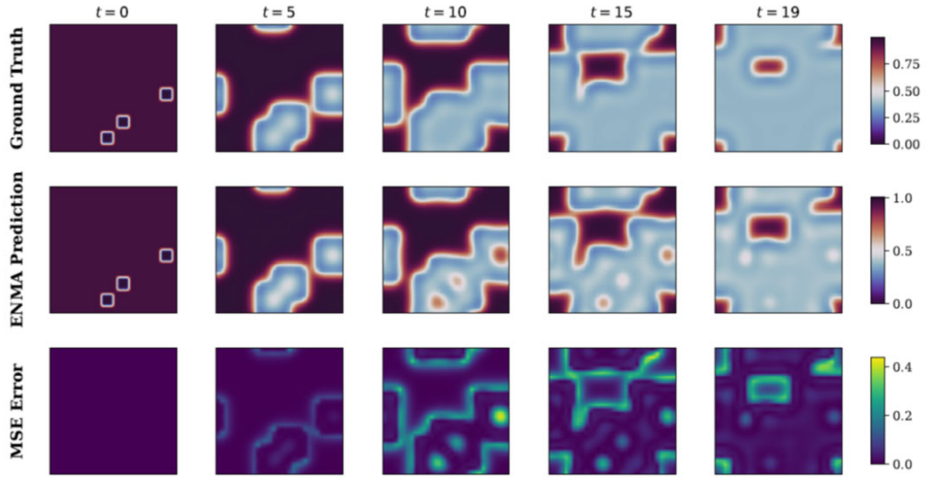


Figure 39: Out-of-distribution (OOD) generalization for the Gray-Scott equation. ENMA prediction remains consistent despite being evaluated at unseen parameters ($F = 0.0467$, $k = 0.058$).

ENMA: Tokenwise Autoregression for Generative Neural PDE Operators (Kassai et al. 2025)

Examples: vorticity

In-distribution

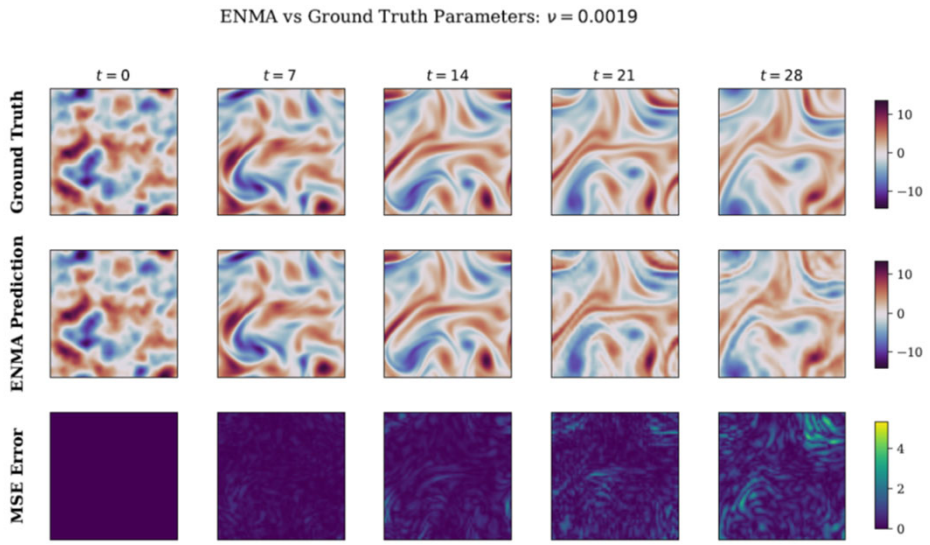


Figure 43: Qualitative comparison between ENMA prediction and ground truth for an in-distribution sample from the Vorticity dataset ($\nu = 0.0019$).

Out-of-distribution

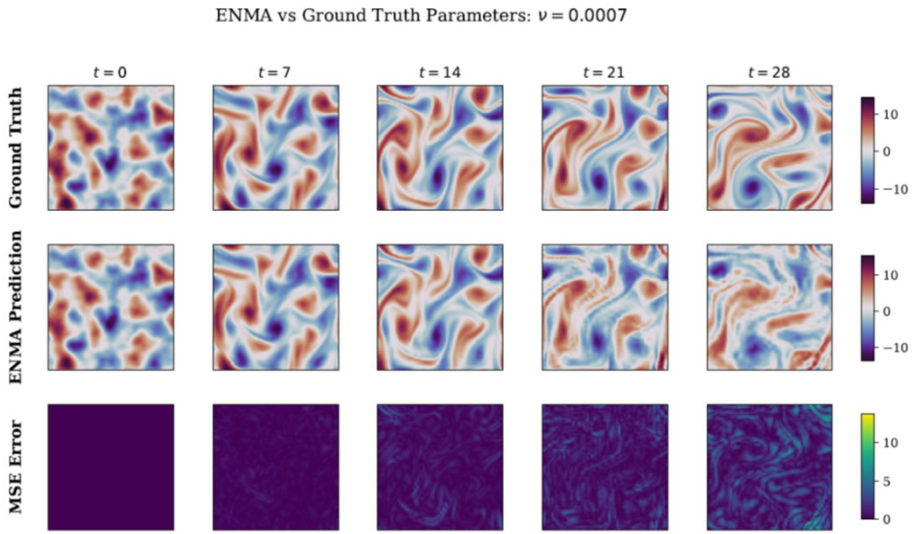


Figure 45: Out-of-distribution example from the Vorticity dataset ($\nu = 0.0007$), highlighting ENMA's robustness in extrapolating vortex dynamics.

Whats next: foundation models

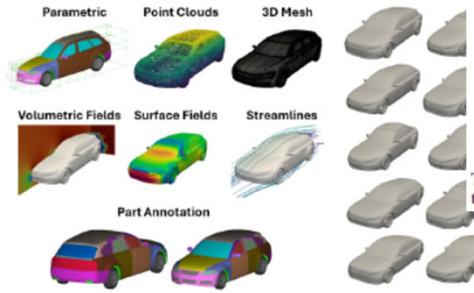
DrivAerNet++: A Large-Scale Multimodal Car Dataset with Computational Fluid Dynamics Simulations and Deep Learning Benchmarks

Mohamed Elrefaie*
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139 USA

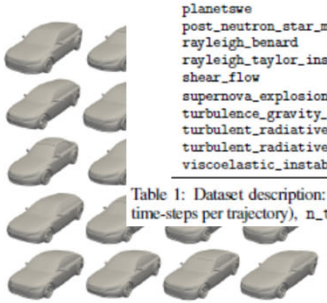
Florin Morar
Morphing and Optimization Solutions
BETA CAE SYSTEMS USA, Inc
Farmington Hills, MI 48334 USA

Angela Dai
Department of Computer Science
Technical University of Munich
Garching, 85748 Germany

Faez Ahm
Department of Mechani
Massachusetts Institute
Cambridge, MA 0



(a) Data modalities of DrivAerNet++. Top row: different data representations; middle row: CFD simulation results; bottom row: annotated car components.



(b) Selected samples from DrivAerNet++ showing diversity in shape with different car designs (fastback, estateback, and notchback), wheels configurations, and underbody configurations.

Figure 1: Data modalities and shape variations in the DrivAerNet++ dataset.

The Well: a Large-Scale Collection of Diverse Physics Simulations for Machine Learning

Ruben Ohana [1,2,*], Michael McCabe [1,*], Lucas Meyer [1], Rudy Morel [1,2], Fruzsina J. Agocs [2,3,†], Miguel Beneitez [4,†], Marsha Berger [2,5,†], Blakesley Burkhart [2,6,†], Stuart B. Dalziel [4,†], Drummond B. Fielding [2,7,†], Daniel Fortunato [2,†], Jared A. Goldberg [2,†], Keiya Hirashima [1,2,8,†], Yan-Fei Jiang [2,†], Rich R. Kerswell [4,†], Suryanarayana Maddu [2,†], Jonah Miller [9,†], Payel Mukhopadhyay [10,†], Stefan S. Nixon [4,†], Jeff Shen [11,†], Romain Watteaux [12,†], Bruno Régaldo-Saint Blancard [1,2], François Rozet [1,13], Liam H. Parker [1,2,10], Miles Cranmer [1,4], Shirley Ho [1,2,5,11]

Dataset	CS	Resolution (pixels)	n_steps	n_tra
acoustic_scattering	(x,y)	256×256	100	800
active_matter	(x,y)	256×256	81	36
convective_envelope_rsg	(r,θ,ϕ)	$256\times128\times256$	100	2
euler_multi_quadrants	(x,y)	512×512	100	10000
gray_scott_reaction_diffusion	(x,y)	128×128	1001	1200
helmholtz_staircase	(x,y)	1024×256	50	512
MHD	(x,y,z)	64^3 and 256^3	100	100
planetswe	(θ,ϕ)	256×512	1008	120
post_neutron_star_merger	$(\log r,\theta,\phi)$	$192\times128\times66$	181	8
rayleigh_benard	(x,y)	512×128	200	1750
rayleigh_taylor_instability	(x,y,z)	$128\times128\times128$	120	45
shear_flow	(x,y)	128×256	200	1120
supernova_explosion	(x,y,z)	64^3 and 128^3	59	1000
turbulence_gravity_cooling	(x,y,z)	$64\times64\times64$	50	2700
turbulent_radiative_layer_2D	(x,y)	128×384	101	90
turbulent_radiative_layer_3D	(x,y,z)	$128\times128\times256$	101	90
viscoelastic_instability	(x,y)	512×512	variable	260

Table 1: Dataset description: coordinate system (CS), resolution of snapshots, n_steps (number of time-steps per trajectory), n_traj (total number of trajectories in the dataset).

APEBench: A Benchmark for Autoregressive Neural Emulators of PDEs

Felix Koehler
Technical University of Munich
Munich Center for Machine Learning
f.koehler@tum.de

Simon Niedermayr
Technical University of Munich
simon.niedermayr@tum.de

Rüdiger Westermann
Technical University of Munich

Nils Thuerey
Technical University of Munich

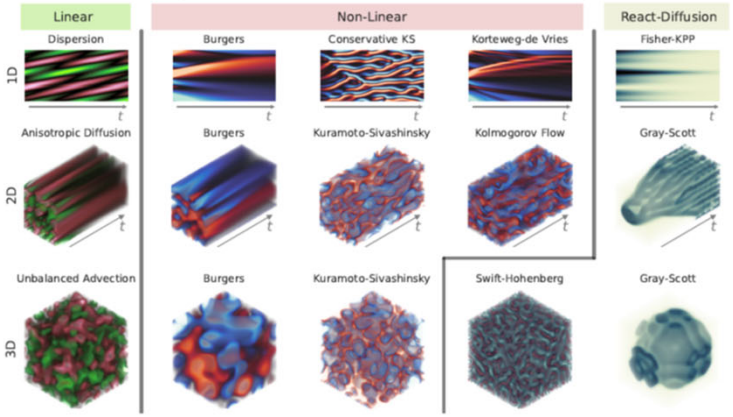


Figure 1: APEBench provides an efficient pseudo-spectral solver to simulate 46 PDE dynamics across one to three spatial dimensions. Shown are examples visualized with APEBench's custom volume renderer.

Whats next: foundation models

Poseidon, Herde et al 2024

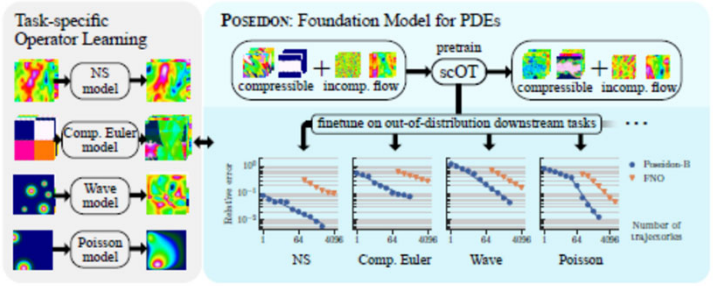
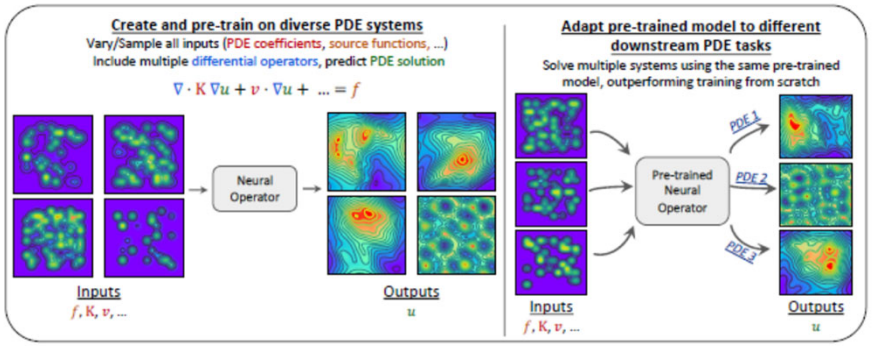
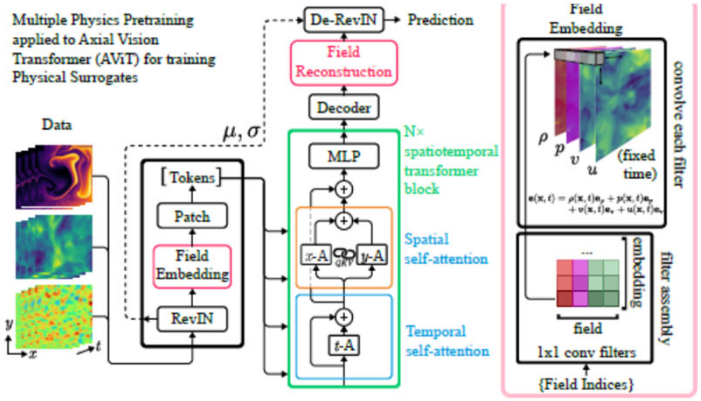


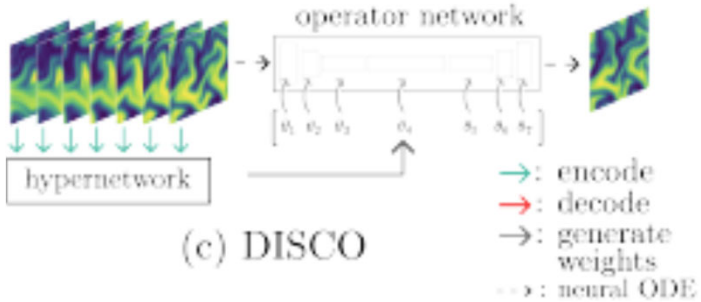
Figure 1: As opposed to PDE-specific operator learning, our pretrained model POSEIDON is up to multiple orders of magnitude more sample efficient than a task-specific neural operator while also being able to transfer to unseen physics during finetuning.



Subramanian et al 2024

MPP, McCabe et al 2024





DISCO, Morel et al 2024

Summing up

- NN generalization for physics
  - Critical problem: deserves specific approaches
  - Proposed approach: adaptive conditioning
    - pro: improves over ERM/ Fine tuning
    - cons: reality gap still to explore
  - Extends to other situations, e.g. multiple physics, multi fidelity, in-context continuous distributions, etc

Thank You