Generalizing to New Physical Systems via Context-Informed Dynamics Model

Matthieu Kirchmeyer*[1,2], Yuan Yin*[1], Jérémie Donà[1], Nicolas Baskiotis[1], Alain Rakotomamonjy[2,3], Patrick Gallinari[1,2]

*Equal Contribution [1]Sorbonne Université, ISIR, MLIA [2]Criteo AI Lab [3]Université de Rouen, LITIS

{matthieu.kirchmeyer,yuan.yin}@sorbonne-universite.fr

Motivation

- Neural dynamics models successful for modeling a physical system but fail on out-of-distribution systems.
- Limitation for real-world problems:
  - *predicting disease diffusion in new countries*
  - *modelling heart blood flow for new patients*
  - *predicting ocean dynamics for new spatial regions on earth*
- *CoDA* (Context-Informed Dynamics Adaptation): first principled solution to this problem.

Problem Setting

Dynamics-aware formulation

We consider dynamical systems described by a differential equation (ODE/PDE):

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = f(x(t))$$

- $x(t)$: state at t
- f: unknown dynamics describing the evolution of x
  - *depends on context e.g. parameters, forcing*
  - *defines trajectories:* $x(t) = x_0 + \int_0^t f(x(\tau))\mathrm{d}\tau$

Learning across environments

We learn a neural dynamics model g_θ across contexts.

- We leverage several different environments:
  - *environment* $e \in \mathcal{E} \Leftrightarrow$ *physical context*
  - *trajectories* $\mathcal{D}^e$ *of corresponding dynamics* f^e
- **Training**: environments $\mathcal{E}_{\mathrm{tr}}$ with *reasonable* data
- **Adaptation**: environments $\mathcal{E}_{\mathrm{ad}}$ with *scarce* data
- **Task**: accurately predict new trajectories of $\mathcal{E}_{\mathrm{ad}}$

Theoretical Motivation

Proposition 1 (Low-rank gradients). *For linearly parameterized dynamics with d_p parameters,* $\forall \theta^c \in \mathbb{R}^{d_\theta}$, $\dim(\mathrm{Span}(\{\nabla_\theta \mathcal{L}(\theta^c, \mathcal{D}^e)\}_{e\in\mathcal{E}})) \le d_p \ll d_\theta$.

→ *Gradients of MSE loss $\mathcal{L}(\theta, \mathcal{D}^e)$ across environments live in a tiny subspace.*

CoDA Framework

Adaptation rule

$$\forall e, \theta^e \triangleq \theta^c + \delta\theta^e$$

θ^c shared; $\delta\theta^e$ environment-specific parameters $\in \mathbb{R}^{d_\theta}$

Locality

$$\min_{\theta^c, \{\delta\theta^e\}} \sum_{e\in\mathcal{E}} \|\delta\theta^e\|^2 \text{ s.t. } \forall t \; \frac{\mathrm{d}x^e(t)}{\mathrm{d}t} = g_{\theta^c+\delta\theta^e}(x^e(t))$$

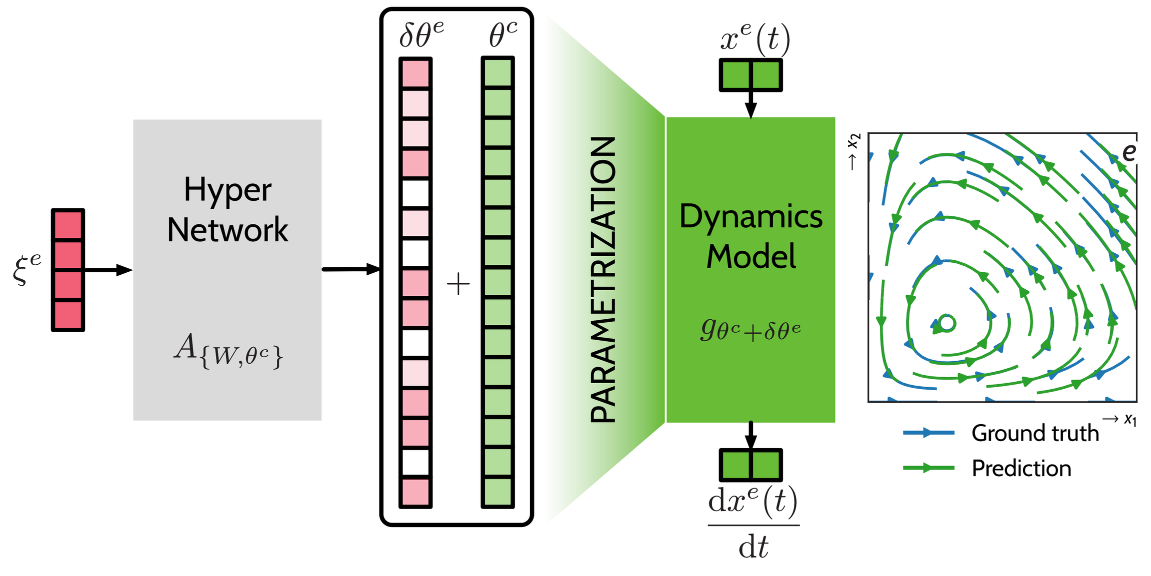
- Fast adaptation by constraining θ^c during training
  → *few update steps*
- Hypothesis space constrained around θ^c
  → *under assumptions, optimization is quadratic and convex*

Low-rank adaptation

- θ^e generated via a trained hypernetwork:

$$\forall e, \theta^e \triangleq \theta^c + W\xi^e \quad (\delta\theta^e \triangleq W\xi^e)$$

- ξ^e: context vector $\in \mathbb{R}^{d_\xi}$ ($d_\xi \ll d_\theta$)
  → *smaller adaptation space*

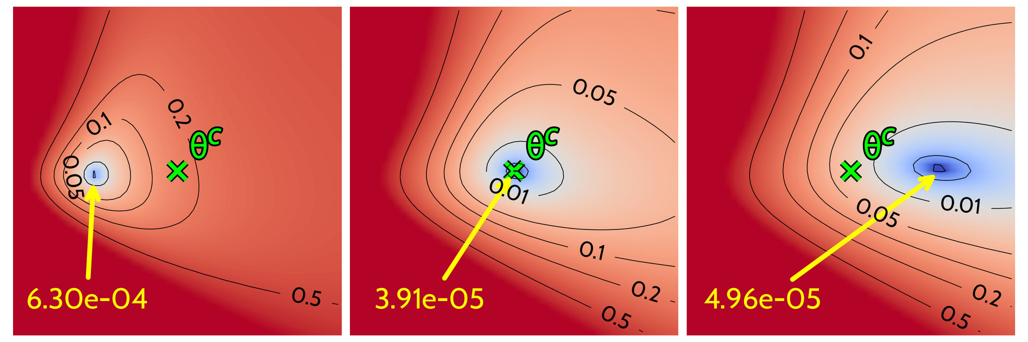


Benefits

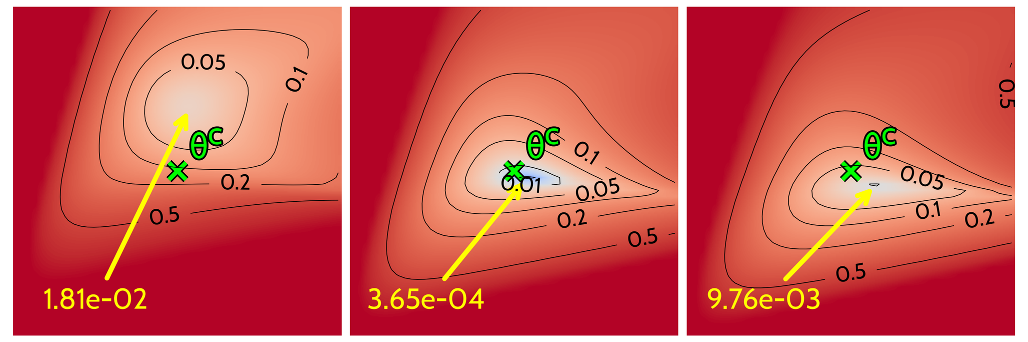
- Fast and sample-efficient adaptation
- Time-continuous
- Agnostic to the architecture of g_θ:
  ODE → MLP PDE → ConvNet, FNO
- Efficient task conditioning: hypernet decoding generalizes FiLM / Concatenation conditioning approaches

Loss Landscapes

CoDA's loss landscape on LV projected onto $\mathrm{Span}(W_1, W_2)$



ERM's loss landscape on LV projected onto top two SVD directions



- Smooth loss with single minimum across environments
- Proximity of local loss optima to θ^c
- Lower minimal loss value for *CoDA* than ERM

Implementation

Objective function

Training: $\min_{\theta^c, W, \{\xi^e\}_{e\in\mathcal{E}_{\mathrm{tr}}}} \sum_{e\in\mathcal{E}_{\mathrm{tr}}} \mathcal{L}(\theta^c + W\xi^e, \mathcal{D}^e) + \|W\xi^e\|^2$

Adaptation: $\min_{\{\xi^e\}_{e\in\mathcal{E}_{\mathrm{ad}}}} \sum_{e\in\mathcal{E}_{\mathrm{ad}}} \mathcal{L}(\theta^c + W\xi^e, \mathcal{D}^e) + \|W\xi^e\|^2$

- Loss: $\mathcal{L}(\theta, \mathcal{D}^e) = \sum_{i=1}^{N_{tr}} \sum_{t_k} \|(x^{e,i} - \tilde{x}^{e,i})(t_k)\|_2^2$
  where $\tilde{x}^{e,i}(t_k) = x_0^{e,i} + \int_0^{t_k} g_\theta(\tilde{x}^{e,i}(\tau))\mathrm{d}\tau$
- Regularization: $\|W\xi^e\|^2 \rightarrow \lambda_\xi\|\xi^e\|_2^2 + \lambda_\Omega \Omega(W)$
  ℓ_2: $\Omega(W) = \|W\|_2$ $\quad\ell_1$: $\Omega(W) = \sum_{i=1}^{d_\theta} \|W_{i,\cdot}\|_2$

Experimental Setting

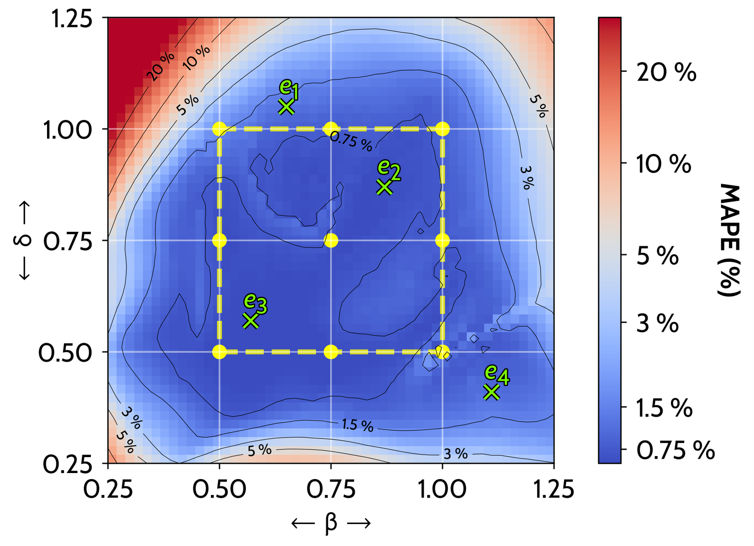
- ODE: Lotka-Volterra (LV), Glycolitic Oscillator (GO)
- PDE: Gray-Scott (GS), Navier-Stokes (NS)
- d_p parameters vary between physical systems ($d_p = 2$: LV, GO, GS; $d_p = 1$: NS)

Results

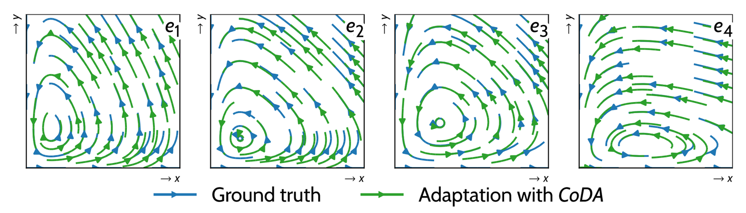
Evaluation on new test trajectories: In-Domain ($\mathcal{E}_{\mathrm{tr}}$) and 1-shot Adaptation ($\mathcal{E}_{\mathrm{ad}}$, $N_{ad} = 1$)

Test MSE (↓)	LV ($\times 10^{-5}$)		GO ($\times 10^{-4}$)		GS ($\times 10^{-3}$)		NS ($\times 10^{-4}$)	
	In-Domain	Adaptation	In-Domain	Adaptation	In-Domain	Adaptation	In-Domain	Adaptation
MAML	60.3±1.3	3150±940	57.3±2.1	1081±62	3.67±0.53	2.25±0.39	68.0±8.0	51.1±4.0
LEADS	3.70±0.27	47.61±12.47	31.4±3.3	113.8±41.5	2.90±0.76	1.36±0.43	14.00±1.55	28.60±7.23
CAVIA-FiLM	4.38±1.15	8.41±3.20	4.44±1.46	3.87±1.28	2.81±1.15	1.43±1.07	23.2±12.1	22.60±9.88
CAVIA-Concat	2.43±0.66	6.26±0.77	5.09±0.35	2.37±0.23	2.67±0.48	1.62±0.85	25.50±6.31	26.00±8.24
CoDA-ℓ_2	1.52±0.08	1.82±0.24	2.45±0.38	1.98±0.06	1.01±0.15	0.77±0.10	9.40±1.13	10.30±1.48
CoDA-ℓ_1	**1.35±0.22**	**1.24±0.20**	**2.20±0.26**	**1.86±0.29**	**0.900±0.057**	**0.74±0.10**	**8.35±1.71**	**9.65±1.37**

Adaptation MAPE (↓) with *CoDA* on LV (• $\mathcal{E}_{\mathrm{tr}}$)

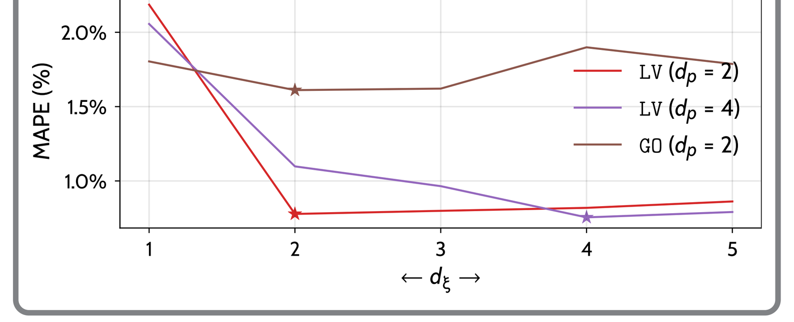


Predicted phase space portraits on LV



LV $\quad \mathrm{d}x/\mathrm{d}t = \alpha x - \beta xy \quad x$: prey
$\quad \mathrm{d}y/\mathrm{d}t = \delta xy - \gamma y \quad y$: predator

Context's dimension vs *In-Domain* MAPE (↓)



Impact of number of adaptation trajectories N_{ad} on *Adaptation* MSE $\times 10^{-5}$ (↓) on LV

Test MSE	1	5	10
MAML	3150±940	239±16	173±10
LEADS	47.61±12.47	19.89±7.23	19.42±3.52
CoDA-ℓ_1	**1.24±0.20**	**1.21±0.18**	**1.20±0.17**

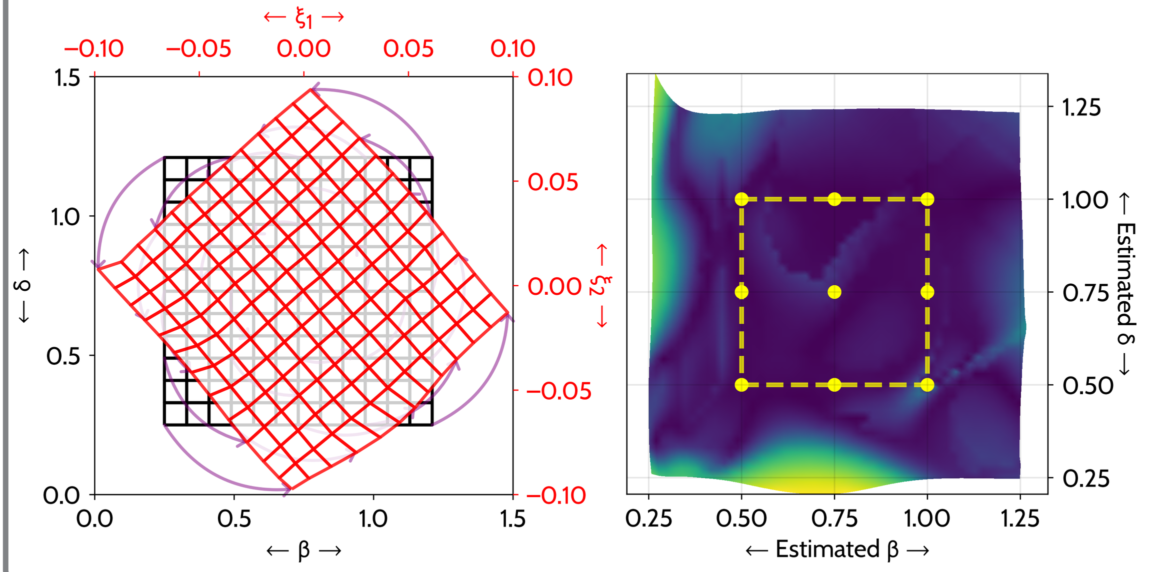
ℓ_2 locality and layer importance for *In-Domain* MSE (↓)

In-Domain MSE (↓)	LV ($\times 10^{-5}$)		GO ($\times 10^{-4}$)	
	W/o ℓ_2	With ℓ_2	W/o ℓ_2	With ℓ_2
Full	2.28±0.29	1.52±0.08	2.98±0.71	2.45±0.38
FirstLayer	2.25±0.29	2.41±0.23	2.38±0.71	**2.12±0.55**
LastLayer	1.86±0.24	**1.27±0.03**	28.40±0.60	28.40±0.64

System Parameter Estimation

- Learn correspondence → between context vectors ⊞ and *known* system parameters ⊞ on $\mathcal{E}_{\mathrm{tr}}$
- Apply the correspondence to $\mathcal{E}_{\mathrm{ad}}$ to infer *unknown* system parameters

LV - correspondence (left); estimation MAPE (↓, right) (• $\mathcal{E}_{\mathrm{tr}}$)



NS - estimation MAPE (↓) (• $\mathcal{E}_{\mathrm{tr}}$)