SCIENCES SORBONNE UNIVERSITÉ

Generalizing to New Physical Systems via Context-Informed Dynamics Model

ICML 2022

July 17th–23rd, 2022

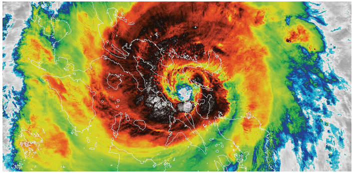
Matthieu Kirchmeyer[⋆1,2], **Yuan Yin**[⋆1],

Jérémie Donà[1], Nicolas Baskiotis[1], Alain Rakotomamonjy[2,3], Patrick Gallinari[1,2]

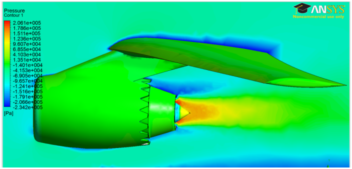
[1]*Sorbonne Université - MLIA ISIR*, [2]*Criteo AI Lab*, [3]*Université de Rouen - LITIS*

ISIR INSTITUT DES SYSTÈMES INTELLIGENTS ET DE ROBOTIQUE

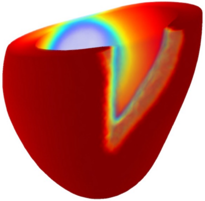
MLIA Machine Learning & Deep Learning for Information Access



Weather prediction

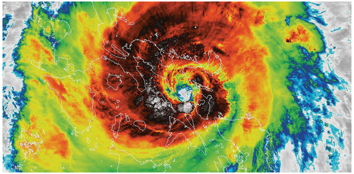


Airplane design

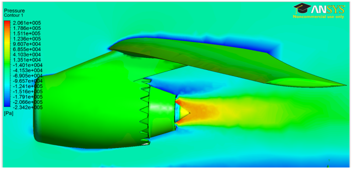


Heart dynamics

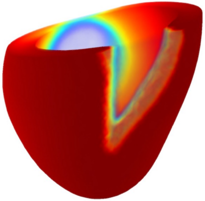
Modelling dynamical systems



Weather prediction



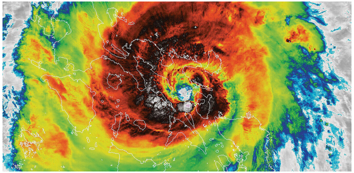
Airplane design



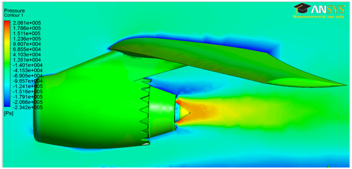
Heart dynamics

Modelling dynamics from data with NNs

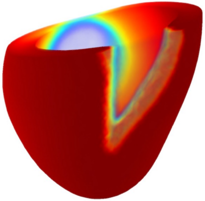
➜ Strong alternative to *using a physical model*.



Weather prediction



Airplane design



Heart dynamics

Modelling dynamics from data with NNs

- ➜ Strong alternative to *using a physical model*.
- ➜ Successfully applied to various problems (Li et al., 2021; Sirignano and Spiliopoulos, 2018; de Bézenac et al., 2018).

Li et al., *Fourier Neural Operator for Parametric Partial Differential Equations. ICLR, 2021*

Sirignano and Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations. Journal of Computational Physics, 2018*

de Bézenac et al., *Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge. ICLR, 2018*

NNs and OOD generalization

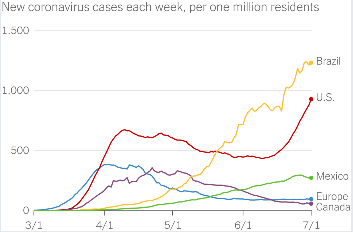
➜ NNs generalize poorly out-of-distribution.

NNs and OOD generalization

- ➜ NNs generalize poorly out-of-distribution.
- ➜ Real-world dynamics models should generalize to new physical contexts:

NNs and OOD generalization

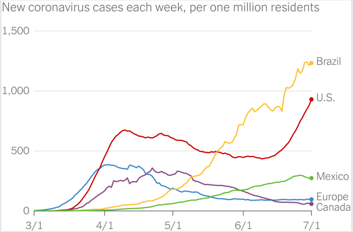
- ➜ NNs generalize poorly out-of-distribution.
- ➜ Real-world dynamics models should generalize to new physical contexts:



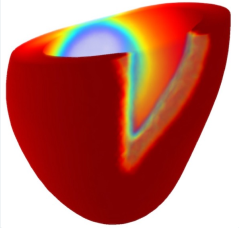
Disease diffusion across countries

NNs and OOD generalization

- ➜ NNs generalize poorly out-of-distribution.
- ➜ Real-world dynamics models should generalize to new physical contexts:



Disease diffusion across countries

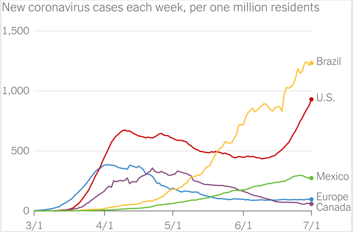


Heart dynamics across patients

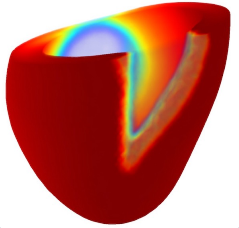
SCIENCES SORBONNE UNIVERSITÉ

NNs and OOD generalization

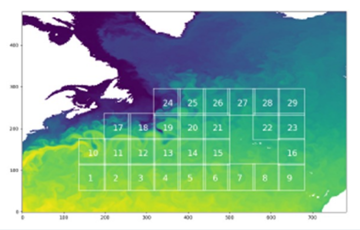
- ➜ NNs generalize poorly out-of-distribution.
- ➜ Real-world dynamics models should generalize to new physical contexts:



Disease diffusion across countries



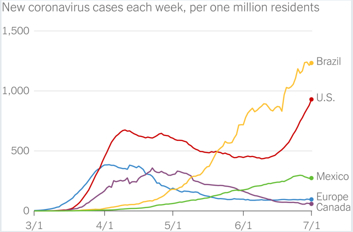
Heart dynamics across patients



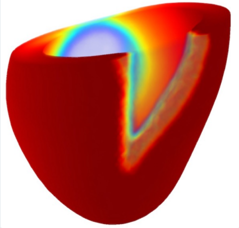
Sea surface temperature across spatial regions

NNs and OOD generalization

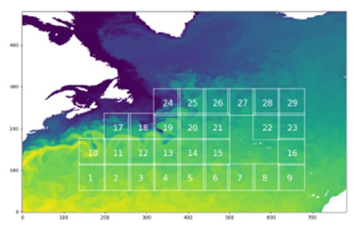
- ➜ NNs generalize poorly out-of-distribution.
- ➜ Real-world dynamics models should generalize to new physical contexts:



Disease diffusion across countries



Heart dynamics across patients



Sea surface temperature across spatial regions

➜ Context-Informed Dynamics Adaptation (CoDA)

➥ one of the first principled solution to this open generalization problem.

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

➜ $x(t)$: state value at t.

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

- ➜ $x(t)$: state value at t.
- ➜ f: *unknown* dynamics describing the evolution of $x(t)$.

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

- ➜ $x(t)$: state value at t.
- ➜ f: *unknown* dynamics describing the evolution of $x(t)$.
  - ➥ defined by a *physical context*: system parameters, external forces...

Formalization

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

- ➜ $x(t)$: state value at t.
- ➜ f: *unknown* dynamics describing the evolution of $x(t)$.
  - ➥ defined by a *physical context*: system parameters, external forces...

Problem setting

- ➜ We learn a *neural dynamics model* $g_\theta \sim f$ across physical contexts.

Formalization

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

- ➜ $x(t)$: state value at t.
- ➜ f: *unknown* dynamics describing the evolution of $x(t)$.
  - ➥ defined by a *physical context*: system parameters, external forces...

Problem setting

- ➜ We learn a *neural dynamics model* $g_\theta \sim f$ across physical contexts.
- ➜ We leverage several different environments:

Formalization

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

- ➜ $x(t)$: state value at t.
- ➜ f: *unknown* dynamics describing the evolution of $x(t)$.
  - ➥ defined by a *physical context*: system parameters, external forces...

Problem setting

- ➜ We learn a *neural dynamics model* $g_\theta \sim f$ across physical contexts.
- ➜ We leverage several different environments:
  - ➥ environment $e \in \mathcal{E} \leftrightarrow$ physical context.

SCIENCES SORBONNE UNIVERSITÉ

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

➜ $x(t)$: state value at t.

➜ f: *unknown* dynamics describing the evolution of $x(t)$.
  ➥ defined by a *physical context*: system parameters, external forces...

Problem setting

➜ We learn a *neural dynamics model* $g_\theta \sim f$ across physical contexts.

➜ We leverage several different environments:
  ➥ environment $e \in \mathcal{E} \leftrightarrow$ physical context.
  ➥ several trajectories of f^e.

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

- ➜ $x(t)$: state value at t.
- ➜ f: *unknown* dynamics describing the evolution of $x(t)$.
  - ➥ defined by a *physical context*: system parameters, external forces...

Problem setting

- ➜ We learn a *neural dynamics model* $g_\theta \sim f$ across physical contexts.
- ➜ We leverage several different environments:
  - ➥ environment $e \in \mathcal{E} \leftrightarrow$ physical context.
  - ➥ several trajectories of f^e.
- ➜ **Training**: environments $\mathcal{E}_{\text{tr}}$ with *reasonable data*.

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

- ➜ $x(t)$: state value at t.
- ➜ f: *unknown* dynamics describing the evolution of $x(t)$.
  - ➥ defined by a *physical context*: system parameters, external forces...

Problem setting

- ➜ We learn a *neural dynamics model* $g_\theta \sim f$ across physical contexts.
- ➜ We leverage several different environments:
  - ➥ environment $e \in \mathcal{E} \leftrightarrow$ physical context.
  - ➥ several trajectories of f^e.
- ➜ **Training**: environments $\mathcal{E}_{\text{tr}}$ with *reasonable data*.
- ➜ **Test**: new adaptation environments $\mathcal{E}_{\text{ad}}$ with *scarce data*.

Notation

Consider physical systems driven by *unknown* differential equations of the form:

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

- ➜ $x(t)$: state value at t.
- ➜ f: *unknown* dynamics describing the evolution of $x(t)$.
  - ➥ defined by a *physical context*: system parameters, external forces...

Problem setting

- ➜ We learn a *neural dynamics model* $g_\theta \sim f$ across physical contexts.
- ➜ We leverage several different environments:
  - ➥ environment $e \in \mathcal{E} \leftrightarrow$ physical context.
  - ➥ several trajectories of f^e.
- ➜ **Training**: environments $\mathcal{E}_{\text{tr}}$ with *reasonable data*.
- ➜ **Test**: new adaptation environments $\mathcal{E}_{\text{ad}}$ with *scarce data*.
- ➜ **Task**: forecast with g_θ new trajectories in adaptation environments $\mathcal{E}_{\text{ad}}$.

Adaptation rule

Splits parameters of g_θ into shared and environment specific parameters:

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

- ➜ θ^c: shared parameters.
- ➜ $\delta\theta^e$: environment-specific parameters.

Adaptation rule

Splits parameters of g_θ into shared and environment specific parameters:

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

- ➜ θ^c: shared parameters.
- ➜ $\delta\theta^e$: environment-specific parameters.

Two principles:

Adaptation rule

Splits parameters of g_θ into shared and environment specific parameters:

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

- ➜ θ^c: shared parameters.
- ➜ $\delta\theta^e$: environment-specific parameters.

Two principles:

Locality.

SCIENCES SORBONNE UNIVERSITÉ

Adaptation rule

Splits parameters of g_θ into shared and environment specific parameters:

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

- ➜ θ^c: shared parameters.
- ➜ $\delta\theta^e$: environment-specific parameters.

Two principles:

Locality.

Fast adaptation to new systems.

Adaptation rule

Splits parameters of g_θ into shared and environment specific parameters:

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

- ➜ θ^c: shared parameters.
- ➜ $\delta\theta^e$: environment-specific parameters.

Two principles:

Locality.	Fast adaptation to new systems.
Low-rank adaptation.	

Adaptation rule

Splits parameters of g_θ into shared and environment specific parameters:

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

- ➜ θ^c: shared parameters.
- ➜ $\delta\theta^e$: environment-specific parameters.

Two principles:

Locality.	Fast adaptation to new systems.
Low-rank adaptation.	Low-dimensionality of the context.

SCIENCES SORBONNE UNIVERSITÉ

Locality constraint

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

Locality constraint

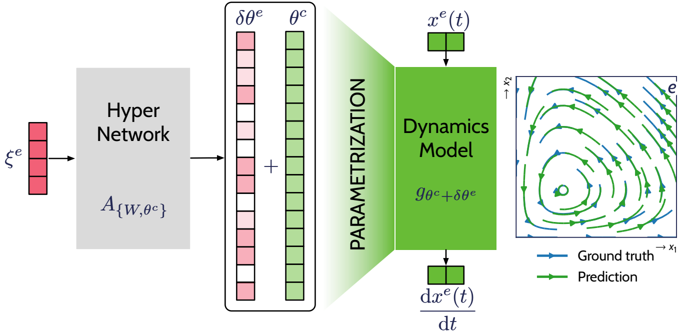
$$\min_{\theta^c, \{\delta\theta^e\}_{e\in\mathcal{E}}} \sum_{e\in\mathcal{E}} \|\delta\theta^e\|^2 \quad \text{subject to} \quad \forall x^e(t), \frac{\mathrm{d}x^e(t)}{\mathrm{d}t} = g_{\theta^c+\delta\theta^e}(x^e(t))$$

- ➜ Fast adaptation by acting as a constraint over θ^c during training
  - ➥ few update steps.
- ➜ Hypothesis space is constrained around θ^c
  - ➥ under assumptions, simplifies optimization into a quadratic convex problem.

Low-rank adaptation

Generate $\delta\theta^e$ with a linear **hypernet** from a learned context vector $\xi^e \in \mathbb{R}^{d_\xi}$:

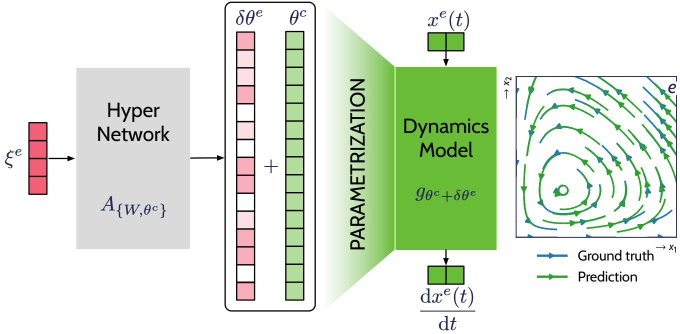
$$\theta^e \triangleq \theta^c + \delta\theta^e \triangleq \theta^c + W\xi^e$$



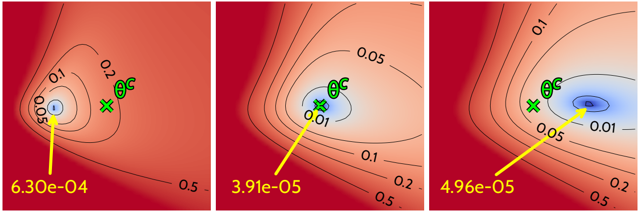
Low-rank adaptation

Generate $\delta\theta^e$ with a linear **hypernet** from a learned context vector $\xi^e \in \mathbb{R}^{d_\xi}$:

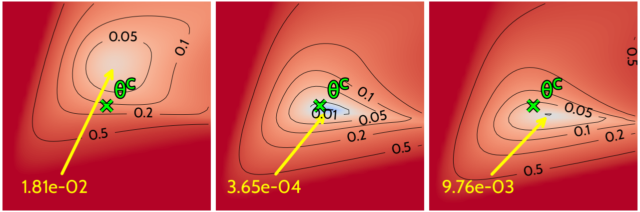
$$\theta^e \triangleq \theta^c + \delta\theta^e \triangleq \theta^c + W\xi^e$$



- ➜ Adaptation in a small subspace: $d_\xi \ll d_\theta$
  - ➥ typically d_ξ is several orders of magnitude smaller than d_θ.
- ➜ ξ^e is adapted to each system with fixed $\{\theta^c, W\}$.



CoDA's loss projected onto subspace $\mathcal{W} \triangleq \mathrm{Span}(W_1, W_2)$.



ERM's loss projected onto the Span of the two principal gradient directions computed with SVD.

Figure 1: Loss landscapes centered in θ^c, marked with ×, for 3 environments on the *Lotka-Volterra* ODE. $\forall e$, $\rightarrow$ points to the local optimum $\theta^{e\star}$ with loss value reported in yellow.

Experimental setting

- Different dynamical systems (2 ODE systems and 2 PDE systems).
- Different architecture for dynamics model g_θ (MLP, Conv, FNO).
- Forecasting new trajectories In-Domain ($\mathcal{E}_{\mathrm{tr}}$) and 1-shot Adaptation ($\mathcal{E}_{\mathrm{ad}}$).

SCIENCES SORBONNE UNIVERSITÉ

Experimental setting

- ➜ Different dynamical systems (2 ODE systems and 2 PDE systems).
- ➜ Different architecture for dynamics model g_θ (MLP, Conv, FNO).
- ➜ Forecasting new trajectories In-Domain ($\mathcal{E}_{\text{tr}}$) and 1-shot Adaptation ($\mathcal{E}_{\text{ad}}$).

Table 1: Test MSE (↓) in training environments $\mathcal{E}_{\text{tr}}$ (*In-Domain*), new environments $\mathcal{E}_{\text{ad}}$ (*Adaptation*). Best in **bold**; Second underlined.

Type	Method	*Lotka-Volterra* $\times 10^{-5}$	*Glycolitic-Oscillator* $\times 10^{-4}$	*Gray-Scott* $\times 10^{-3}$	*Navier-Stokes* $\times 10^{-4}$
		In-domain	In-domain	In-domain	In-domain
GBML	MAML	60.3±1.3	57.3±2.1	3.67±0.53	68.0±8.0
GBML	ANIL	381±76	74.5±11.5	5.01±0.80	61.7±4.3
GBML	Meta-SGD	32.7±12.6	42.3±6.9	2.85±0.54	53.9±28.1
MTL	LEADS	3.70±0.27	31.4±3.3	2.90±0.76	14.0±1.55
Context-based	CAVIA-FiLM	4.38±1.15	4.44±1.46	2.81±1.15	23.2±12.1
Context-based	CAVIA-Concat	2.43±0.66	5.09±0.35	2.67±0.48	25.5±6.31
Context-based	**CoDA-ℓ_2**	<u>1.52±0.08</u>	<u>2.45±0.38</u>	<u>1.01±0.15</u>	<u>9.40±1.13</u>
Context-based	**CoDA-ℓ_1**	**1.35±0.22**	**2.20±0.26**	**0.90±0.057**	**8.35±1.71**

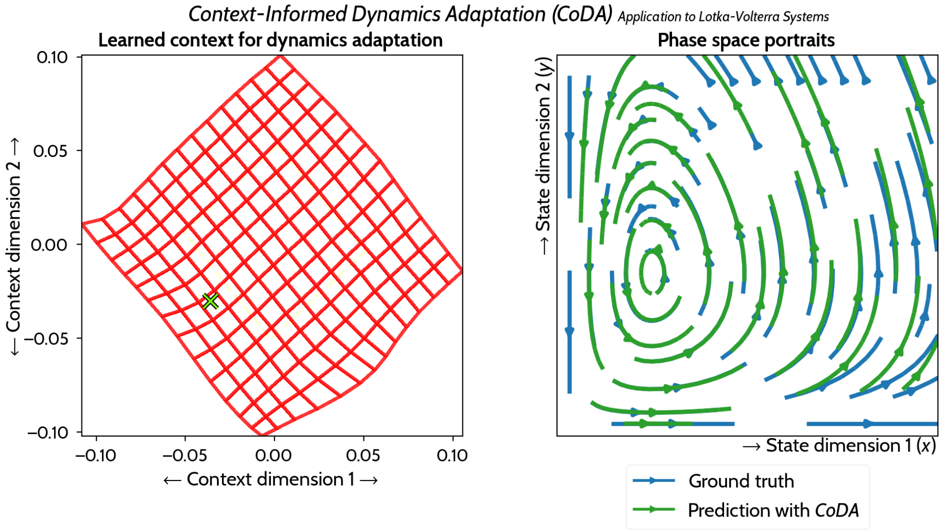
Experimental setting

- ➜ Different dynamical systems (2 ODE systems and 2 PDE systems).
- ➜ Different architecture for dynamics model g_θ (MLP, Conv, FNO).
- ➜ Forecasting new trajectories In-Domain ($\mathcal{E}_{\mathrm{tr}}$) and 1-shot Adaptation ($\mathcal{E}_{\mathrm{ad}}$).

Table 1: Test MSE (↓) in training environments $\mathcal{E}_{\mathrm{tr}}$ (*In-Domain*), new environments $\mathcal{E}_{\mathrm{ad}}$ (*Adaptation*). Best in **bold**; Second <u>underlined</u>.

Type	Method	*Lotka-Volterra* $\times 10^{-5}$		*Glycolitic-Oscillator* $\times 10^{-4}$		*Gray-Scott* $\times 10^{-3}$		*Navier-Stokes* $\times 10^{-4}$	
		In-domain	Adaptation	In-domain	Adaptation	In-domain	Adaptation	In-domain	Adaptation
GBML	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
	ANIL	381±76	4570±2390	74.5±11.5	1688±226	5.01±0.80	3.95±0.11	61.7±4.3	48.6±3.2
	Meta-SGD	32.7±12.6	7220±4580	42.3±6.9	1573±413	2.85±0.54	2.68±0.20	53.9±28.1	44.3±27.1
MTL	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	14.0±1.55	28.6±7.23
Context-based	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	22.6±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	25.5±6.31	26.0±8.24
	CoDA-ℓ_2	<u>1.52±0.08</u>	<u>1.82±0.24</u>	<u>2.45±0.38</u>	<u>1.98±0.06</u>	<u>1.01±0.15</u>	<u>0.77±0.10</u>	<u>9.40±1.13</u>	<u>10.3±1.48</u>
	CoDA-ℓ_1	**1.35±0.22**	**1.24±0.20**	**2.20±0.26**	**1.86±0.29**	**0.90±0.057**	**0.74±0.10**	**8.35±1.71**	**9.65±1.37**

Adaptation with CoDA in action



Context is closely linked to system parameters.

SCIENCES SORBONNE UNIVERSITÉ

Take-home messages

- ➜ Local and low-rank principles simplify the optimization and allow achieving fast, flexible and sample-efficient adaptation.
- ➜ The framework is agnostic to the dynamics model. Feel free to change to other parameterized models.
- ➜ For dynamics, CoDA generalizes better than existing meta-learning / multi-task learning baselines.

SCIENCES SORBONNE UNIVERSITÉ

Take-home messages

- ➜ Local and low-rank principles simplify the optimization and allow achieving fast, flexible and sample-efficient adaptation.
- ➜ The framework is agnostic to the dynamics model. Feel free to change to other parameterized models.
- ➜ For dynamics, CoDA generalizes better than existing meta-learning / multi-task learning baselines.

Check out more in the paper

- ➜ Can accurately predict parameters of new *unknown* physical systems.
- ➜ Theoretical results supporting the framework.

Paper `arxiv.org/abs/2202.01889` **Code** `github.com/yuan-yin/CoDA`

Contact `{matthieu.kirchmeyer,yuan.yin}@isir.upmc.fr`