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]

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

MLIA Machine Learning & Deep Learning for Information Access

CRITEO

Modelling dynamical systems

Weather forecasting

Airplane design

Heart dynamics

Modelling dynamical systems

Weather forecasting

Airplane design

Heart dynamics

Modelling dynamics from data with NNs

Ü Strong + flexible alternative to *physical models*.

Modelling dynamical systems

Weather forecasting

Airplane design

Heart dynamics

Modelling dynamics from data with NNs

- Ü Strong + flexible alternative to *physical models*.
- Ü 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

Ü NNs generalize poorly out-of-distribution.

NNs and OOD generalization

- Ü NNs generalize poorly out-of-distribution.
- Ü Limitation for real-world dynamics models, e.g. when modelling:

NNs and OOD generalization

- Ü NNs generalize poorly out-of-distribution.
- Ü Limitation for real-world dynamics models, e.g. when modelling:

Disease diffusion across countries

NNs and OOD generalization

- Ü NNs generalize poorly out-of-distribution.
- Ü Limitation for real-world dynamics models, e.g. when modelling:

Disease diffusion across countries

Heart dynamics across patients

NNs and OOD generalization

- Ü NNs generalize poorly out-of-distribution.
- Ü Limitation for real-world dynamics models, e.g. when modelling:

Disease diffusion across countries

Heart dynamics across patients

Sea surface temperature across spatial regions

Generalization and dynamical systems

NNs and OOD generalization

- Ü NNs generalize poorly out-of-distribution.
- Ü Limitation for real-world dynamics models, e.g. when modelling:

Disease diffusion across countries

Heart dynamics across patients

Sea surface temperature across spatial regions

Ü Context-Informed Dynamics Adaptation (CoDA)
å one of the first principled solution to this open generalization problem.

Formalization

Notation and objective

Physical systems driven by *unknown* differential equations:

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

Formalization

Notation and objective

Physical systems driven by *unknown* differential equations:

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

Ü $x(t)$: state value at t.

Formalization

Notation and objective

Physical systems driven by unknown differential equations:

$$\frac{dx(t)}{dt} = f(x(t))$$

- $x(t)$: state value at t.
- f: unknown dynamics.

Formalization

Notation and objective

Physical systems driven by *unknown* differential equations:

$$\frac{dx(t)}{dt} = f(x(t))$$

- $x(t)$: state value at t.
- f: unknown dynamics.
  - defined by a physical context: system parameters, external forces...

Formalization

Notation and objective

Physical systems driven by unknown differential equations:

$$\frac{dx(t)}{dt} = f(x(t))$$

- $x(t)$: state value at t.
- f: unknown dynamics.
  - defined by a physical context (system parameters, external forces...)

Goal: Learn dynamics across contexts with a neural dynamics model $\hat{f}$.

Formalization

Notation and objective

Physical systems driven by unknown differential equations:

$$\frac{dx(t)}{dt} = f(x(t))$$

- $x(t)$: state value at t.
- f: unknown dynamics.
  - defined by a physical context (system parameters, external forces...)

Goal: Learn dynamics across contexts with a neural dynamics model $\hat{f}$.

Multi-environment learning problem

Formalization

Notation and objective

Physical systems driven by unknown differential equations:

$$\frac{dx(t)}{dt} = f(x(t))$$

- $x(t)$: state value at t.
- f: unknown dynamics.
  - defined by a physical context (system parameters, external forces...)

Goal: Learn dynamics across contexts with a neural dynamics model $\hat{f}_\theta$.

Multi-environment learning problem

- Environment e:

Formalization

Notation and objective

Physical systems driven by unknown differential equations:

$$\frac{dx(t)}{dt} = f(x(t))$$

- $x(t)$: state value at t.
- f: unknown dynamics.
  - defined by a physical context e: system parameters, external forces...

Goal: Learn dynamics across contexts with a neural dynamics model $\hat{f}_\theta$.

Multi-environment learning problem

- Environment e:
  - physical context.

Formalization

Notation and objective

Physical systems driven by unknown differential equations:

$$\frac{dx(t)}{dt} = f(x(t))$$

- $x(t)$: state value at t.
- f: unknown dynamics.
  - defined by a physical context c: system parameters, external forces...

Goal: Learn dynamics across contexts with a neural dynamics model $\hat{f}_\theta$.

Multi-environment learning problem

- Environment e:
  - physical context.
  - several observed trajectories of f^e.

Formalization

Notation and objective

Physical systems driven by unknown differential equations:

$$\frac{dx(t)}{dt} = f(x(t))$$

- $x(t)$: state value at t.
- f: unknown dynamics.
  - defined by a physical context e: system parameters, external forces...

Goal: Learn dynamics across contexts with a neural dynamics model $\hat{f}_\theta$.

Multi-environment learning problem

- Environment e:
  - physical context.
  - several observed trajectories of f^e.
- Training: environments E_{tr} with reasonable data

Formalization

Notation and objective

Physical systems driven by unknown differential equations:

$$\frac{dx(t)}{dt} = f(x(t))$$

- $x(t)$: state value at t.
- f: unknown dynamics.
  - defined by a physical context (system parameters, external forces...)

Goal: Learn dynamics across contexts with a neural dynamics model $\hat{f}_\theta$.

Multi-environment learning problem

- Environment e:
  - physical context.
  - several observed trajectories of f^e.
- Training: environments E_{tr} with reasonable data
- Adaptation: generalize to new environments E_{ad} with few data.

CoDA learns

CoDA learns

- Ü shared parameters across environments^{c},.

CoDA Overview

CoDA learns

- Ü shared parameters across environments, ^{c}.
- Ü environment-specific parameters, ^{e}.

CoDA Overview

CoDA learns

- Ü shared parameters across environments, ^{c}.
- Ü environment-specific parameters, ^{e}.
- Ü ^{e}, ^{c} + ^{e}.

CoDA learns

- Ü shared parameters across environments, ^{c}.
- Ü environment-specific parameters, ^{e}.
- Ü ^{e}, ^{c} + ^{e}.

Under two constraints:

CoDA learns

- Ü shared parameters across environments, ^{c}.
- Ü environment-specific parameters, ^{e}.
- Ü ^{e}, ^{c} + ^{e}.

Under two constraints:

Locality.

CoDA Overview

CoDA learns

- Ü shared parameters across environments, ^{c}.
- Ü environment-specific parameters, ^{e}.
- Ü ^{e}, ^{c} + ^{e}.

Under two constraints:

Locality.

Fast adaptation to new systems.

CoDA learns

- Ü shared parameters across environments, ^{c}.
- Ü environment-specific parameters, ^{e}.
- Ü ^{e}, ^{c} + ^{e}.

Under two constraints:

Locality.

Fast adaptation to new systems.

Low-rank adaptation.

CoDA learns

- Ü shared parameters across environments, ^{c}.
- Ü environment-specific parameters, ^{e}.
- Ü ^{e}, ^{c} + ^{e}.

Under two constraints:

Locality.

Fast adaptation to new systems.

Low-rank adaptation.

Low-dimensionality of the context.

Constrained optimization problem

$$\min_{c;f\ {}^{e}g_{e2E}} \sum_{e2E} k\ {}^{e}k^{2} \quad \text{s.t.} \quad 8x^{e}(t); \frac{dx^{e}(t)}{dt} = g_{c+}\ {}_{e}(x^{e}(t))$$

Constrained optimization problem

$$\min_{c;f\ {}^{e}g_{e2E}} \sum_{e2E} k\ {}^{e}k^{2} \quad \text{s.t.} \quad 8x^{e}(t); \frac{dx^{e}(t)}{dt} = g\ {}_{c+}\ {}_{e}(x^{e}(t))$$

Ü Fast adaptation.

^{e} generated via a linea hypernetwork $A_{f\,W;\ ^{c}g}$:

$$^{e}, \quad A_{f\,W;\ ^{c}g}(\ ^{e}) = \ ^{c} + W\ ^{e}$$

^{e} generated via a linea hypernetwork $A_{f\,W;\ ^{c}g}$:

$$^{e}, \quad A_{f\,W;\ ^{c}g}(\ ^{e}) = \ ^{c} + W\ ^{e}$$

Ü ^{e}: context vector - low-dimensional, environment-specific

^{e} generated via a linea hypernetwork $A_{f\,W;\ ^{c}g}$:

$$^{e}, \quad A_{f\,W;\ ^{c}g}(\ ^{e}) = \ ^{c} + W\ ^{e}$$

Ü ^{e}: context vector - low-dimensional, environment-specific

Ü Rows of W : shared adaptation directions

^{e} generated via a linear hypernetwork $A_{f\,W;\ ^{c}g}$:

$$^{e}, \quad A_{f\,W;\ ^{c}g}(\ ^{e}) = \ ^{c} + W\ ^{e}$$

Ü ^{e}: context vector - low-dimensional, environment-specific

Ü Rows of W: shared adaptation directions

å Fixed low-dimensional adaptation subspace W, $\mathrm{Span}(W_1; \ ; W_{\dim(\ )})$

^{e} generated via a linear hypernetwork $A_{f\,W;\ ^{c}g}$:

$$^{e}, \quad A_{f\,W;\ ^{c}g}(\ ^{e}) = \ ^{c} + W\ ^{e}$$

Ü ^{e}: context vector - low-dimensional, environment-specific

Ü Rows of W: shared adaptation directions

å Fixed low-dimensional adaptation subspace W, $\mathrm{Span}(W_1; \ ; W_{\dim(\ )})$

Ü Adaptation is parameter efficient.

^{e} generated via a linear hypernetwork $A_{f\,W;\ ^{c}g}$:

$$^{e}, \quad A_{f\,W;\ ^{c}g}(\ ^{e}) = \ ^{c} + W\ ^{e}$$

Ü ^{e}: context vector - low-dimensional, environment-specific

Ü Rows of W: shared adaptation directions

å Fixed low-dimensional adaptation subspace ^{e}W, $\mathrm{Span}(W_1; \ ; W_{\dim(\ )})$

Ü Adaptation is parameter efficient.

Ü Experimentally sample-efficient.

Loss landscape

Figure 1: CoDA's projected loss landscape onto W for 3 *Lotka-Volterra* systems.

Loss landscape

Figure 1: CoDA's projected loss landscape onto W for 3 *Lotka-Volterra* systems.

Low-rank

Ü Adaptation subspace W contains optima (!) with low loss.

Loss landscape

Figure 1: CoDA's projected loss landscape onto W for 3 *Lotka-Volterra* systems.

Low-rank

Ü Adaptation subspace W contains optima (!) with low loss.

Locality

Ü Proximity of optima (!) to ^{c} (×).

Quantitative results

Experimental setting

Ü Datasets: ODEs + PDEs with *unknown* parameters that vary across systems.

Table 1: MSE on new test trajectories (#). Best in **bold**; Second underlined.

	Lotka-Volterra 10^{5}	*Glycolitic-Oscillator* 10^{4}	*Gray-Scott* 10^{3}	*Navier-Stokes* 10^{4}
8 < :				
8 ≪ w ≪				

Quantitative results

Experimental setting

- Ü Datasets: ODEs + PDEs with *unknown* parameters that vary across systems.
- Ü Dynamics-aware baselines based on meta-learning; multi-task learning.

Table 1: MSE on new test trajectories (#). Best in **bold**; Second underlined.

	Method	*Lotka-Volterra* 10^{5}	*Glycolitic-Oscillator* 10^{4}	*Gray-Scott* 10^{3}	*Navier-Stokes* 10^{4}
GBML	MAML				
	ANIL				
	Meta-SGD				
MTL	LEADS				
Context-ual	CAVIA-FiLM				
	CAVIA-Concat				
	CoDA-$`_2$				
	CoDA-$`_1$				

Quantitative results

Experimental setting

- Datasets: ODEs + PDEs with *unknown* parameters that vary across systems.
- Dynamics-aware baselines based on meta-learning; multi-task learning.
- Evaluation:
  - In-Domain (E_{tr}).

Table 1: MSE on new test trajectories (#). Best in **bold**; Second underlined.

	Method	*Lotka-Volterra* 10^{-5}	*Glycolitic-Oscillator* 10^{-4}	*Gray-Scott* 10^{-3}	*Navier-Stokes* 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
	ANIL	381±76	74.5±11.5	5.01±0.80	61.7±4.3
	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-aware	CAVIA-FiLM	4.38±1.15	4.44±1.46	2.81±1.15	23.2±12.1
	CAVIA-Concat	2.43±0.66	5.09±0.35	2.67±0.48	25.5±6.31
	CoDA-ℓ_2	1.52±0.08	2.45±0.38	1.01±0.15	9.40±1.13
	CoDA-ℓ_1	**1.35±0.22**	**2.20±0.26**	**0.90±0.057**	**8.35±1.71**

Quantitative results

Experimental setting

- Datasets: ODEs + PDEs with *unknown* parameters that vary across systems.
- Dynamics-aware baselines based on meta-learning; multi-task learning.
- Evaluation:
  - In-Domain (E_{tr}).
  - 1-shot Adaptation (E_{ad}).

Table 1: MSE on new test trajectories (#). Best in **bold**; Second underlined.

	Method	*Lotka-Volterra* 10^{5}		*Glycolitic-Oscillator* 10^{4}		*Gray-Scott* 10^{3}		*Navier-Stokes* 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-aware	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	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	10.3±1.48
	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**

Conclusion

Take-home messages

Ü New SoTA approach to handle OOD generalization in dynamical systems.

Conclusion

Take-home messages

- Ü New SoTA approach to handle OOD generalization in dynamical systems.
- Ü Fast, parameter-efficient and sample-efficient adaptation.

Conclusion

Take-home messages

- Ü New SoTA approach to handle OOD generalization in dynamical systems.
- Ü Fast, parameter-efficient and sample-efficient adaptation.

Paper arxiv.org/abs/2202.01889
Code github.com/yuan-yin/CoDA
Contact {matthieu.kirchmeyer,yuan.yin}@isir.upmc.fr

Conclusion

Take-home messages

- Ü New SoTA approach to handle OOD generalization in dynamical systems.
- Ü Fast, parameter-efficient and sample-efficient adaptation.

Paper arxiv.org/abs/2202.01889
Code github.com/yuan-yin/CoDA
Contact {matthieu.kirchmeyer,yuan.yin}@isir.upmc.fr

Check out our poster: Hall E #313 19/07 - 5:30 p.m. — 7:30 p.m !