
Environment Inference for Learning Generalizable Dynamical System

Shixuan Liu¹ Yue He^{2*} Haotian Wang¹ Wenjing Yang¹ Yunfei Wang³
Peng Cui^{4*} Zhong Liu⁵

¹College of Computer Science and Technology, National University of Defense Technology

²School of Information, Renmin University of China

³College of Systems Engineering, National University of Defense Technology

⁴Department of Computer Science and Technology, Tsinghua University

⁵Laboratory for Big Data and Decision, National University of Defense Technology

szftandy@hotmail.com, hy865865@gmail.com

{wanghaotian13, wenjing.yang, wangyunfei, liuzhong}@nudt.edu.cn

cui@tsinghua.edu.cn

Abstract

Data-driven methods offer efficient and robust solutions for analyzing complex dynamical systems but rely on the assumption of I.I.D. data, driving the development of generalization techniques for handling environmental differences. These techniques, however, are limited by their dependence on environment labels, which are often unavailable during training due to data acquisition challenges, privacy concerns, and environmental variability, particularly in large public datasets and privacy-sensitive domains. In response, we propose DynaInfer, a novel method that infers environment specifications by analyzing prediction errors from fixed neural networks within each training round, enabling environment assignments directly from data. We prove our algorithm effectively solves the alternating optimization problem in unlabeled scenarios and validate it through extensive experiments across diverse dynamical systems. Results show that DynaInfer outperforms existing environment assignment techniques, converges rapidly to true labels, and even achieves superior performance when environment labels are available.

1 Introduction

Data-driven approaches, especially neural networks, offer a powerful alternative or complement to traditional physics-based methods for understanding complex dynamical systems [4]. Neural network-based emulators are particularly valuable for their ability to provide fast, cost-effective approximations of complex simulations [9, 22], making them especially useful in scenarios where the underlying physics are poorly understood or misinterpreted, or where external disturbances are difficult to model [43, 34]. These emulators are adept at handling large sets of variables and solving problems that are challenging for conventional solvers. Recent advancements in deep learning, along with innovative methods for modeling temporal and spatio-temporal systems, have led to a significant increase in applications across various fields, ranging from simple Hamiltonian dynamics to more complex areas like fluid dynamics and climatology [32, 7].

While recent advancements have shown promising results, they often rely on the assumption that abundant, static data are available to satisfy the independent and identically distributed (IID) hypothesis. However, this assumption is frequently violated in practice due to challenges in data collection, associated costs, and environmental changes driven by exogenous factors [24, 26]. Recent work

*Corresponding authors

in dynamical systems addresses this by introducing a multi-environment setting, where trajectories follow distinct dynamics across environments. These studies developed generalization methods that learn a shared global component while accounting for environment-specific variations, avoiding the limitations of underperforming averaged models [43, 17].

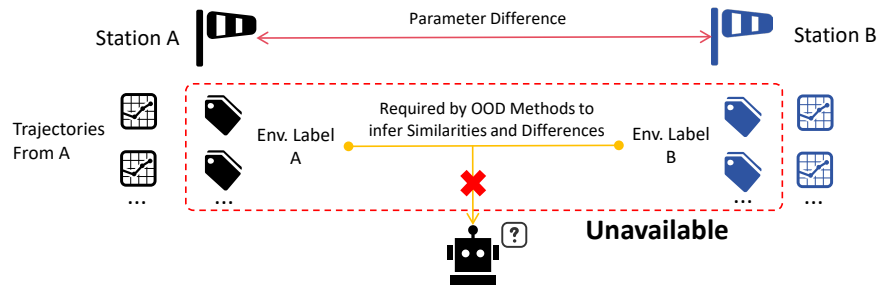


Figure 1: Environment labels, required by current generalization methods, are often unavailable.

Nevertheless, a key limitation of many generalization techniques is their reliance on partitioning datasets across distinct domains or environments, which are assumed to capture underlying variations. These environment labels enable algorithms to identify and exploit both similarities and differences across environments. However, obtaining such environment labels during training is often challenging due to data acquisition difficulties or privacy constraints. For example, in scientific research, data may be collected over time under uncontrolled or unknown conditions [42]. In ecological studies, critical environmental parameters such as temperature or rainfall may vary unpredictably or remain unrecorded. [2]. Similarly, when aggregating data from multiple sources, environment labels are frequently lost or omitted, a common issue in large public datasets [35]. Furthermore, in privacy-sensitive domains like healthcare, finance, or social networking, access to environment-specific information is often restricted [18]. These limitations highlight the need for generalization methods that do not depend on explicit environment labels.

To address the challenge of unknown environment labels, we propose a novel approach that infers environment specifications by leveraging the key insight that trajectories within the same environment share consistent dynamics and exhibit similar prediction losses under the same neural network. This inherent consistency enables us to automatically derive meaningful environment assignments directly from the training data. We introduce an environment inference objective designed for dynamical systems, which minimizes environment-specific prediction losses. Using fixed neural networks, we first infer environments and then iteratively refine these networks with the inferred environments, ultimately learning a generalizable dynamical system.

Our model identifies environment labels directly from mixed trajectories of dynamical systems, facilitating the training of off-the-shelf generalization algorithms in scenarios where such labels are absent. Importantly, our findings demonstrate that inferring environments from mixed sequence data can improve the performance of generalization strategies, even compared to cases where environments are manually assigned.

Our main contributions are as follows:

- We present the first investigation into the challenge of unlabeled environment conditions in the context of learning generalizable dynamical systems, and propose a general framework named DynaInfer that utilizes the prediction loss to accurately infer latent environment labels from mixed sequence data².
- We theoretically establish that our algorithm effectively solves the alternating optimization problem without requiring environment labels, demonstrating its capacity to discern heterogeneous environments and infer generalizable mechanisms.
- We examine the efficacy of DynaInfer through experiments in both in-domain settings and adaptation scenarios using three representative dynamical systems. Results confirm that the environment labels assigned by DynaInfer converge rapidly to the true labels.

²Code is available at <https://github.com/shixuanliu-andy/DynaInfer>

The remainder of this paper is structured as follows. Section 2 clarifies the problem definition. Section 3 introduces our framework and provides the theoretical underpinnings. Section 4 details the experimental setup and discusses the results. Related work is reviewed in Section 5, and Section 6 concludes the paper.

2 Problem Definition

2.1 Dynamical Systems

We examine dynamical systems determined by unidentified differential equations evolving over time, expressed as,

$$\frac{dx_t}{dt} = f(x_t) \quad (1)$$

where $t \in \mathbb{R}$ is the time index within a time interval $I = [0, T]$, and x_t is a time-variant state within a bounded set $\mathcal{A}$. The evolution function $f : \mathcal{A} \rightarrow T\mathcal{A}$ maps x_t to its temporal derivative in the tangent space $T\mathcal{A}$ and belongs to a class of vector fields $\mathcal{F}$.

In this paper, we consider both ordinary differential equation (ODE) and partial differential equation (PDE). For ODEs, $\mathcal{A} \subset \mathbb{R}^d$; for PDEs, $\mathcal{A}$ represents a d' -dimensional vector field within a bounded spatial domain (such as 2D or 3D Euclidean space) denoted as $S \subset \mathbb{R}^{d'}$. The function f characterizes the data distribution of trajectories $\mathcal{T}$. Trajectories initiated from $x_0 \sim p(X_0)$ are computed by integrating the derivatives: $x_t = x_0 + \int_0^t f(x_u) du, \forall t \in I$.

2.2 Multi-Environment Dynamical Systems Learning

In contrast to the standard expected risk minimization (ERM) framework, which assumes i.i.d. trajectories, the multi-environment learning problem involves learning trajectories from M different environments. In each environment $e \in [M] = \{1, 2, \dots, M\}$, the trajectories are governed by unique differential equations described by function f_e . Specifically, consider N trajectories $\{x^1, x^2, \dots, x^N\}$, where each trajectory x^i is associated with an environment $e_i \in [M]$. The dynamics of each trajectory x^i are thus modeled by the differential equation $dx_t^i/dt = f_{e_i}(x_t^i)$. The set of environments for all trajectories is denoted by $e = \{e_1, e_2, \dots, e_N\} \in [M]^N$.

In multi-environment learning, the goal is to enhance traditional ERM methods by exploiting both the commonalities and disparities across diverse environments. To this end, the dynamics is decomposed into two components: a global component shared across all environments, parameterized by θ , and an environment-specific component, parameterized by ϕ_e for each environment e . The set of environment-specific parameters is denoted by $\phi = \{\phi_e\}_{e \in [M]}$. Consequently, the dynamics of each trajectory x^i are parameterized by both the universal and environment-specific parameters,

$$\frac{dx_t^i}{dt} = h(x_t^i; \theta, \phi_{e_i}).$$

This parametrization entails a decomposition that can be implemented either functionally or parametrically. The functional decomposition, expressed as $h(x_t^i; \theta, \phi_e) = f_\theta(x_t^i) + g_{\phi_e}(x_t^i)$, distinguishes between a shared function f_θ and an environment-specific function g_{ϕ_e} [43]. Alternatively, the parametric decomposition integrates the environment-specific parameters directly, formulated as $h(x_t^i; \theta, \phi_e) = f_{\theta+\phi_e}(x_t^i)$ [17]. Intuitively, the key ingredient for multi-environment learning is that θ should encapsulate the maximal shared dynamics, whereas ϕ_e should exclusively reflect the unique characteristics of each environment e not described by θ . However, directly optimizing both parameters poses an ill-posed problem, often resulting in trivial solutions where the global component learns nothing meaningful. To counteract this, the regularization term $\Omega(\phi_e)$ is introduced to effectively penalize ϕ_e , thereby facilitating learning in the global component. Consequently, with the information about the environments $e = \{e_1, e_2, \dots, e_N\}$, the loss function is given by,

$$R_e(\theta, \phi) = \sum_{i=1}^N \int_{t \in I} \left\| \frac{dx_t^i}{dt} - h(x_t^i; \theta, \phi_{e_i}) \right\|_2^2 dt + \lambda \sum_{e=1}^M \Omega(\phi_e). \quad (2)$$

The first term evaluates the regression precision of the parameterized function $h(\cdot; \theta, \phi_e)$. The ground truth vector field (VF) is not explicitly known and derived from trajectory data. Using the learned VF, a simulated trajectory is generated and used to calculate the regression loss by referring to real trajectories during training. The term $\Omega(\phi_e)$ serves as a regularization term for ϕ_e , with λ controlling the intensity of the regularization.

2.3 Environment Inference for Multi-Environment Learning

In many real-world scenarios, the environment label for a trajectory sample is unknown. We aim to infer an environment assignment for each sample that maximizes the model's generalization ability across different environments. To achieve this goal, we reformulate the learning objective into an optimization problem contingent on a specific environment assignment e . Specifically, our aim is to learn the environment assignment $\hat{e} = \{\hat{e}^1, \hat{e}^2, \dots, \hat{e}^N\} \in [M]^N$ for each trajectory to effectively optimize Equation (2). The overall objective is defined as follows:

$$\hat{e}^*, \theta^*, \phi^* = \arg \min_{\hat{e}, \theta, \phi} R_{\hat{e}}(\theta, \phi). \quad (3)$$

In this paper, we explore a particularly challenging scenario where the total number of training environments M is also unknown. We investigate the development of a practical model that maintains favourable performance even when the exact number of true environments is unknown.

3 The DynaInfer Framework

In this section, we introduce our framework that operates on field functions without prior domain knowledge, proving especially effective in dynamical systems where exogenous factors are unobserved and in situations where relevant environmental information is unclear or absent. While some clustering methods infer labels for CV data, they operate on finite-dimensional vectors in Euclidean space, which drastically differs from field functions, making them inapplicable.

The optimization challenge in Equation (3) is primarily due to the inherently discrete nature of the environment assignments $\hat{e}$, which take values in the set $[M]$. This discrete categorization impedes the direct application of traditional gradient descent methods, which are typically designed for continuous parameter spaces. To effectively address this challenge, we develop a dual iterative strategy that concurrently updates the environment assignments $\hat{e}$ and the model parameters θ, ϕ . The first step in our approach centers on inferring environment labels by analyzing the prediction errors of the trajectories output by the neural network during the current training round. This analysis serves as a diagnostic tool to uncover critical discrepancies that signify distinct dynamical environments. Following this, the second step entails refining the neural network parameters based on the newly inferred environment assignments in an unbiased manner, enabling the neural network to precisely adapt to the unique characteristics of each identified environment. Through this adaptive refinement, our model progressively enhances its accuracy and generalization capability across different dynamic settings. The complete method is detailed in Algorithm 1 and is visually depicted in Figure 2.

Algorithm 1 DynaInfer framework

- 1: **Input:** Randomly initialized $\theta, \phi = \{\phi_e\}_{e \in [M]}$, assumed number of environments M , total rounds T_r
- 2: $\theta^{(0)}, \phi^{(0)} \leftarrow \theta, \phi$
- 3: **for** $r \leftarrow 1$ to T_r **do**
- 4: Update $\hat{e}^{(r)}$ based on Equation (4)
- 5: Update $\theta^{(r)}, \phi^{(r)}$ based on Equation (5)
- 6: **end for**
- 7: **Output:** θ, ϕ .

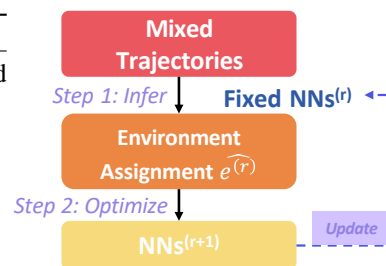


Figure 2: Model Framework.

3.1 Bias-aware Environment Assignment

The environment inference step receives a single dataset as input and generates a partition of the data into multiple environments. Intuitively, trajectories originating from the same environment adhere to

consistent dynamics. Employing the same neural network to model these trajectories should yield similar estimation error across them, reflecting a coherence in their dynamic parameters.

Upon examination, we observe that the optimization framework defined in Equation (2) shares conceptual similarities with classical centroid-based clustering methodologies, although the latter generally operate in Euclidean space. In K-means clustering, the primary goal is to minimize the within-cluster sum of squares, often referred to as cluster inertia [37]. This minimization effort concentrates on reducing the distances between the points within each cluster and their corresponding centroid, which typically converges to a local optimum. This characteristic enables K-means to efficiently delineate distinct and compact clusters, capturing the core essence of data distribution with respect to spatial proximity.

This insight prompts us to explore a conceptual analogy wherein the neural network that minimizes the loss most effectively operates analogously to a "centroid" for a cluster of trajectories within the same dynamic environment. We characterise the distance between a trajectory (data point) and the network (centroid) by the regression loss of the trajectory using the network. Initially, with a randomly initialized network—analogue to a randomly initialized centroid in K-means clustering—we assign each trajectory a label based on the minimal prediction loss calculated from all available networks. Subsequently, we refine this "centroid" by optimizing it to minimize the loss as specified in Equation (2). Through this iterative optimization process, we can achieve the objective stated in Equation (3).

More specifically, at round r , given the fixed network parameters from the previous iteration $\theta^{(r-1)}, \phi^{(r-1)}$, the environment assignment $\hat{e}_i^{(r)}$ is updated through the following process,

$$\hat{e}_i^{(r)} = \arg \min_{e \in [M]} \int_{t \in I} \left\| \frac{dx_t^i}{dt} - h \left(x_t^i; \theta^{(r-1)}, \phi_e^{(r-1)} \right) \right\|_2^2 dt. \quad (4)$$

If multiple solutions exist for Equation (4) and $\hat{e}_i^{(r-1)}$ minimizes it, we retain this assignment for the next round, i.e., $\hat{e}_i^{(r)} = \hat{e}_i^{(r-1)}$. This approach ensures the validity of a constant loss reduction (to be stated in Proposition 3.1).

3.2 Assignment-driven Optimization

After assigning trajectories to specific clusters, we proceed to update the conceptual centroid by optimizing network parameters. In the K-means algorithm, centroids are recalculated by averaging the positions of all points within each cluster. Similarly, our method updates network parameters by considering the mean estimation error over trajectories within a cluster, ensuring unbiased contributions from each trajectory. This approach not only improves the representational accuracy of each cluster but also enables the network to dynamically adapt to the underlying structure of the trajectories, thereby enhancing the efficacy and reliability of our learning process in unlabeled scenarios. Therefore, the parameters $\theta^{(r)}$ and $\phi^{(r)}$ are given by:

$$\theta^{(r)}, \phi^{(r)} = \arg \min_{\theta, \phi} R_{\hat{e}^{(r)}}(\theta, \phi). \quad (5)$$

3.3 Theoretical Property

We begin by demonstrating that Algorithm 1 effectively optimizes Equation (3).

Proposition 3.1. *For all rounds $1 \leq r < T_r$, we must have*

$$R_{\hat{e}^{(r+1)}}(\theta^{(r+1)}, \phi^{(r+1)}) \leq R_{\hat{e}^{(r)}}(\theta^{(r)}, \phi^{(r)}).$$

Furthermore, suppose the space of $\arg \min_{\theta, \phi} R_{\hat{e}}(\theta, \phi)$ is finite for all $\hat{e} \in [M]^N$. Then there exists a constant $C > 0$ such that if $r > 1$ and $R_{\hat{e}^{(r+1)}}(\theta^{(r+1)}, \phi^{(r+1)}) < R_{\hat{e}^{(r)}}(\theta^{(r)}, \phi^{(r)})$, we must have

$$R_{\hat{e}^{(r+1)}}(\theta^{(r+1)}, \phi^{(r+1)}) \leq R_{\hat{e}^{(r)}}(\theta^{(r)}, \phi^{(r)}) - C.$$

Remark 3.1. Given the assumptions made in prior works [43, 17] that $h(\cdot; \theta, \phi_e)$ is linear with respect to θ and ϕ_e , and that $\Omega(\phi_e)$ is strictly convex with respect to ϕ_e , it follows logically that the space of $\arg \min_{\theta, \phi} R_{\hat{e}}(\theta, \phi)$ is finite for all $\hat{e} \in [M]^N$, as is evident from Equation (2). The proof is provided in Appendix A.

This proposition demonstrates that, as long as the loss in consecutive rounds of Algorithm 1 decreases, the loss must decrease by a constant $C > 0$.

4 Experiments

Our experiments investigate three dynamical systems governed by specific differential equations: an ODE for biological modeling, PDEs for reaction-diffusion in chemistry, and the Navier-Stokes equations for incompressible fluid dynamics. These complex, nonlinear systems test our method's ability to classify spatio-temporal patterns and physical laws across diverse environments.

4.1 Environment Specification

We provide a basic introduction to the datasets here, with detailed descriptions in Appendix E. **Lotka-Volterra (LV) [23]** The system models the dynamics between a prey-predator pair in an ecosystem, captured by the following ODE:

$$dm/dt = \alpha m - \beta mn, dn/dt = \delta mn - \gamma n$$

where m and n represent the population densities of the prey and predator, respectively, and α , β , δ , and γ are the interaction parameters between the two species.

Gray-Scott (GS) [28] The model uses simple reaction-diffusion equations to effectively study complex pattern formation in chemical and biological systems, following underlying PDE dynamics:

$$\begin{aligned}\partial m/\partial t &= D_m \Delta m - mn^2 + F(1 - m), \\ \partial n/\partial t &= D_n \Delta n - mn^2 - (F + k)n.\end{aligned}$$

where m and n represent the concentrations of two chemical components in the spatial domain S with periodic boundary conditions; D_m and D_n are their constant diffusion coefficients; and F and k are the reaction parameters that govern the spatio-temporal dynamic patterns.

Navier-Stokes (NS) [22] The Navier-Stokes PDE describes the motion of viscous fluid substances:

$$\partial m/\partial t = -n \nabla m + \nu \Delta m + \xi, \nabla v = 0$$

where n is the velocity field, $m = \nabla \times n$ is the vorticity, both n and m lie in a spatial domain S with periodic boundary conditions, ν is the viscosity (fixed at $1e^{-3}$), and ξ is the constant forcing term in the domain S .

4.2 Experimental Setting and Baselines

Settings. We evaluate DynaInfer in two distinct settings: in-domain generalization on $\mathcal{E}_o$ and adaptation to new environments in $\mathcal{E}_u$, with $\mathcal{E}_o$ and $\mathcal{E}_u$ hosting disjoint environments. For in-domain experiments, both training and testing occur on $\mathcal{E}_o$. At test time, environment labels are also not provided and are instead inferred from the prediction bias over an initial segment of the trajectory (less than $2\Delta t$ for practical reasons). For adaptation experiments, we follow standard domain adaptation practice: initial training on the source domain $\mathcal{E}_o$ is followed by fine-tuning and testing on the target domain $\mathcal{E}_u$, where environment labels are provided.

Dataset Preparation. For in-domain experiments, we generate four LV trajectories in each of nine environments, ten GS trajectories in each of three environments, and eight NS trajectories in each of four environments. For adaptation experiments, we simulate the same number of trajectories per environment, conducting finetuning in two additional environments $e \in \mathcal{E}_u$. All dynamic environment parameters are detailed in Appendix E. For evaluation, we sample 32 trajectories per environment, initialized according to the underlying distribution $p(x_0)$. The LV and GS data are generated using the DOPRI5 solver [8, 12], while the NS data is simulated with the pseudo-spectral method as in [22].

Baselines. We explore three potential strategies for assigning environment labels in the absence of environmental information, compared to our method (DynaInfer): grouping all samples into a single environment (All in One), assigning a distinct environment label to each sample (One per Env), and random assignment (Random). Additionally, we consider an "Oracle" assignment method where labels are fully known during training, bringing the total to five labeling strategies. Furthermore,

we consider three base models for dynamical system generalization: LEADS [43], CoDA- l_1 , and CoDA- l_2 [17]. We utilize the neural network architectures and parameter configurations as described in their papers for each type of dynamic system. By combining these assignment methods with base models, we generate fifteen distinct methods for evaluation. In adaptation experiments, during fine-tuning, LEADS and CoDA adhere to the protocol described in their papers, by fixing the shared components or parameters and rendering only the $\mathcal{E}_u$ -specific components trainable. All neural network architectures, optimizers, and parameters for the base models are configured as described in their respective papers.

Metrics. To rigorously evaluate predictive accuracy in dynamical system learning, we adopt two complementary metrics: Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE), averaged over 5 independent runs.

4.3 Experimental Results

4.3.1 In-domain Generalization Results

The in-domain generalization results detailed in Table 1 illustrate the performance implications of various assignment strategies. We observe that the "All in One" and "Random" assignment strategies consistently underperform across multiple datasets and baseline models. While the "One per Env" strategy yields only mediocre results, it provides a viable initial approach in scenarios where no labels are available. Across all datasets, DynaInfer significantly outperforms other assignment strategies. Furthermore, DynaInfer consistently shows effectiveness across all tested base models and datasets, underscoring its robustness against diverse methods and datasets. Notably, DynaInfer either matches or exceeds Oracle performance, particularly in complex PDE environments like GS and NS, suggesting that its bias-aware approach effectively compensates for not having access to the true labels available to Oracle.

In Figure 3, DynaInfer's predicted states qualitatively align closely with the ground truth and Oracle, occasionally outperforming Oracle (e.g., in GS dataset with LEADS base model, where Oracle shows some jitters).

Data	Assignment	LEADS			CoDA- l_1			CoDA- l_2		
		Train MSE	Test MSE	MAPE	Train MSE	Test MSE	MAPE	Train MSE	Test MSE	MAPE
LV	All in One	7.17 E-2	7.41±0.02 E-2	49.22±1.84	7.14 E-2	7.40±0.01 E-2	49.44±3.15	7.17 E-2	7.41±0.00 E-2	39.26±22.13
	One per Env	4.15 E-4	4.91±3.50 E-4	6.68±2.44	8.68 E-4	9.14±0.41 E-4	5.67±1.01	8.18 E-4	8.43±0.39 E-4	5.73±1.19
	Random	7.20 E-2	7.38±0.02 E-2	50.01±1.05	7.12 E-2	7.39±0.01 E-2	48.87±1.81	7.09 E-2	7.39±0.00 E-2	48.86±2.54
	DynaInfer	4.74 E-5	7.93±2.49 E-5	2.83±1.62	9.57 E-5	1.83±3.40 E-4	3.27±2.36	1.71 E-4	1.82±3.07 E-4	2.02±1.66
	Oracle	4.55 E-5	7.02±0.76 E-5	1.78±0.10	1.78 E-5	3.19±0.24 E-5	1.26±0.06	1.99 E-5	2.72±0.18 E-5	1.21±0.08
GS	All in One	8.73 E-3	9.60±0.02 E-3	3008.80±892.20	9.24 E-3	9.61±0.03 E-3	4115.80±223.54	9.25 E-3	9.60±0.00 E-3	3723.00±713.85
	One per Env	1.38 E-3	1.65±0.54 E-3	173.44±59.16	1.56 E-3	1.91±0.06 E-3	185.23±61.43	1.52 E-3	1.87±0.02 E-3	174.08±57.82
	Random	8.78 E-3	9.36±0.20 E-3	1403.50±119.50	9.25 E-3	9.59±0.03 E-3	3958.25±682.38	8.77 E-3	9.35±0.02 E-3	3919.88±157.54
	DynaInfer	3.60 E-5	4.14±0.21 E-5	117.57±33.90	9.22 E-5	1.23±0.41 E-4	122.93±22.05	6.69 E-5	7.25±2.11 E-5	112.52±14.15
	Oracle	7.73 E-5	1.34±0.76 E-4	97.77±12.09	6.04 E-5	9.60±3.91 E-5	163.38±47.89	4.69 E-5	7.04±1.84 E-5	138.86±16.55
NS	All in One	5.34E-02	6.71±0.11 E-2	239.70±14.78	5.79E-02	6.64±0.11 E-2	251.38±8.54	6.17E-02	6.64±0.03 E-2	255.26±9.11
	One per Env	2.24E-02	4.11±0.14 E-2	169.48±9.68	3.45E-02	3.88±0.22 E-2	161.04±10.73	2.31E-02	4.04±0.22 E-2	158.26±9.35
	Random	3.06E-02	6.58±0.05 E-2	233.80 ± 8.44	5.04E-02	6.58±0.05 E-2	247.95±4.59	5.78E-02	6.66±0.04 E-2	254.47±6.40
	DynaInfer	6.10E-04	7.05±0.34 E-3	77.29±10.18	1.23E-02	1.62±0.18 E-2	108.17±10.30	8.92E-04	1.19±0.12 E-2	96.57±12.75
	Oracle	2.59E-04	6.55±1.34 E-3	67.58±9.37	1.36E-02	1.73±0.29 E-2	124.22±12.35	7.11E-04	9.46±0.51 E-3	91.06±5.85

Table 1: In-domain Experiment Results on the LV, GS, and NS environments. Our approach consistently outperforms all non-oracle assignment methods, and beats oracle at times, demonstrating its effectiveness in modeling heterogeneous environments and generalizing across dynamical systems.

4.3.2 Domain Adaptation Results

The results in Table 2 demonstrate the performance of different assignment strategies under the domain adaptation setting. The "One per Env" strategy consistently outperforms the "All in One" approach. While the "Random" assignment benefits the base model LEADS, it slightly diminishes CoDA's performance across all datasets. DynaInfer shows strong adaptation capabilities across various datasets and base generalization methods, consistently outperforming other non-Oracle techniques. This indicates that DynaInfer effectively captures commonalities across environments, enabling smoother adaptation to new conditions. Furthermore, the performance gap between DynaInfer and the Oracle is significantly narrower in adaptation tasks compared to in-domain generalization.

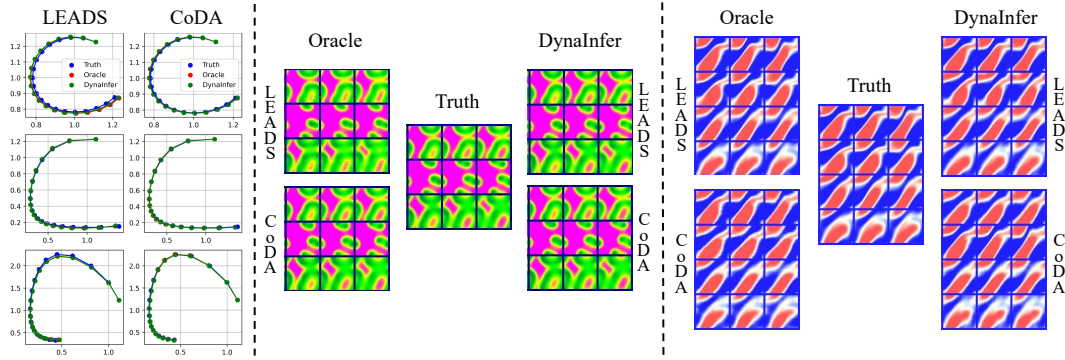


Figure 3: Left: Predicted test trajectories from 3 environments vs. ground truth and Oracle for LV with 2 base models. Middle and Right: Predicted last 3 states for GS and NS, respectively, vs. ground truth and Oracle using 2 base models. Each environment is shown by row. For CoDA, we use the best Oracle result's CoDA version to save space. See Appendix H for all the visualizations.

Data	Assignment	LEADS		CoDA- l_1		CoDA- l_2	
		MSE	MAPE	MSE	MAPE	MSE	MAPE
LV	All in One	4.16±8.61 E-2	9.92±3.55	4.01±6.43 E-2	26.90±10.65	4.10±7.61 E-2	27.80±8.81
	One per Env	2.28±1.81 E-3	5.41±0.50	1.72±0.53 E-3	27.63±7.81	1.66±0.82 E-3	25.51±7.72
	Random	1.72±0.53 E-3	6.87±0.13	1.14±0.61 E-3	27.56±6.36	1.05±0.54 E-3	29.65±6.31
	DynaInfer	5.77±1.46 E-4	2.84±0.13	8.37±0.94 E-5	10.16±0.04	8.49±2.07 E-5	10.30±0.08
	Oracle	1.67±2.26 E-3	3.16±0.37	5.85±1.24 E-5	10.24±0.06	5.12±3.17 E-5	10.24±0.04
GS	All in One	4.59±1.18 E-4	721.25±197.54	2.85±0.55 E-3	6658.20±1651.77	2.76±0.72 E-3	7355.20±335.45
	One per Env	3.59±2.71 E-4	450.20±368.36	1.08±0.82 E-3	6247.34±817.74	1.19±0.97 E-3	5948.57±935.71
	Random	5.73±0.86 E-4	1261.00±979.30	2.92±0.80 E-3	7292.88±312.54	2.84±0.89 E-3	4499.25±844.62
	DynaInfer	1.00±0.32 E-4	378.73±182.12	2.41±0.91 E-4	220.54±65.60	2.13±0.41 E-4	207.96±46.49
	Oracle	2.21±0.93 E-4	434.73±432.38	2.66±0.79 E-4	302.68±188.25	2.10±0.89 E-4	230.62±118.29
NS	All in One	1.25±2.04 E-2	67.17±3.33	1.25±0.20 E-2	218.66±27.26	1.29±0.29 E-2	214.44±17.08
	One per Env	2.78±2.08 E-2	96.23±4.54	2.04±0.68 E-2	214.38±15.19	2.43±0.48 E-2	209.68±18.98
	Random	1.32±0.53 E-2	81.18±7.43	4.66±1.04 E-2	215.21±19.39	4.37±0.99 E-2	191.03±16.38
	DynaInfer	7.52±0.76 E-3	50.93±8.83	9.27±1.81 E-3	101.38±15.77	9.71±2.10 E-3	101.04±16.27
	Oracle	1.16±0.68 E-2	57.35±17.90	7.46±0.72 E-3	154.86±41.00	7.32±0.81 E-3	100.04±24.93

Table 2: Adaptation Experiment Results on the LV, GS, and NS environments. DynaInfer consistently outperforms other non-Oracle methods across all datasets and narrows the performance gap with the Oracle more effectively compared to in-domain generalization.

4.3.3 Assignments Convergence

We illustrate the probability of environment assignments with DynaInfer over training time in Figure 4. Initially, our model may default to random assignments due to unoptimized neural networks. However, the assignments quickly converge to the true labels. Notably, systems with simpler dynamics, like LV compared to NS, enable quicker learning of base generalization methods, resulting in faster convergence of environment assignments.

4.3.4 Performance across Varying Number of Assumed Environments

As the true number of environments ($|\mathcal{E}_o|$) might be unknown, we examine DynaInfer's performance with varying assumed environments M in Figure 5. Our findings show that prior knowledge of the true M is beneficial: performance peaks when the assumed M aligns with the true count. Additionally, our model demonstrates robustness to over-estimations of M . Due to its ability to account for bias, our model effectively identifies trajectories from the same environment and remains robust to an excess of inaccurately trained neural networks, even when the assumed M is too large. The above observations suggest that incrementally increasing the number of environments until peak performance is achieved can be a straightforward way to identify the true M . Lastly, DynaInfer consistently outperforms other non-oracle approaches when M is underestimated, except for the One-per-Env baseline (which requires M equal to the number of trajectories and is computationally infeasible).

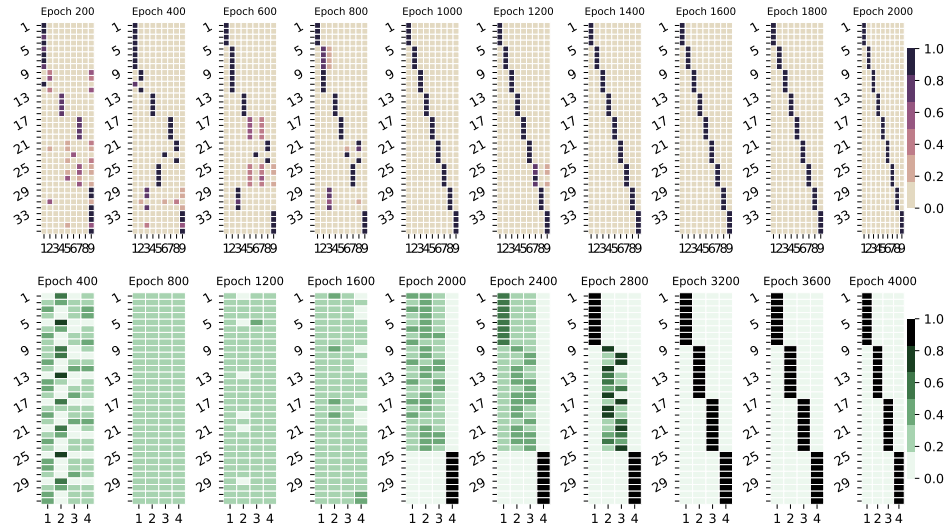


Figure 4: Environment assignment probability over time, averaged over 5 runs, with LEADS as base model (on LV (top) and NS (bottom); see Appendix G for GS). The assignment converges to the true label faster than the designated training steps. A similar trend is observed with the CoDA model.

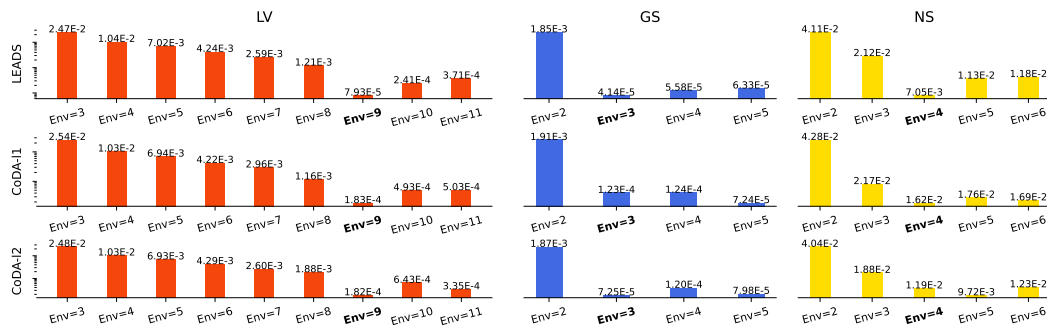


Figure 5: Performance across assumed number of environments with DynaInfer(log-scale y-axis). The peak performance aligns with the true number of environments (bold on x-axis) with high probability, and remains stable thereafter.

5 Related Work

5.1 Domain Generalization and Adaptation

Domain generalization (DG) seeks to train a model on one or multiple distinct but related source domains so that it generalizes effectively to any out-of-distribution (OOD) target domain. DG methods assume data heterogeneity and use additional environment labels to develop models that remain robust across unseen and shifted test data. Many DG strategies focus on domain alignment, aiming to minimize divergence among source domains to achieve domain-invariant representations [21, 14, 29, 31, 40]. Other approaches enhance the diversity of training data by augmenting source domains [5, 33, 39]. Additionally, some methods leverage meta-learning and invariant risk minimization for regularization, further enhancing generalization [20, 1].

Domain adaptation (DA) methods enable model generalization to target domains with shifted data distributions and are primarily classified into three categories. Instance-based methods reweight or adjust training samples to reflect the test distribution [15, 6]. Feature-based approaches align feature distributions across training and test domains [38, 36]. Model-based strategies focus on developing models that are either robust to domain shifts or specifically tailored for the target domain [25, 10].

5.2 Generalization for Dynamical Systems

Generalization in dynamical systems remains underexplored in literature. Among the limited studies, LEADS emerges as a novel multi-task learning framework that effectively generalizes across the functional space of dynamical systems [43]. Alternatively, CoDA optimizes within the parameter space, enhancing model adaptability and efficiency while accommodating increased environmental variability without requiring multiple distinct network trainings for each setting [17]. In contrast, DyAd is a context-aware meta-learning approach that adjusts the dynamics model by decoding a time-invariant context from observed states [41]. Despite its novelty, DyAd relies on potentially impractical weak supervision based on physics-derived quantities and uses Adaptive Instance Normalization, which may degrade performance.

Currently, three notable weaknesses prevail in generalization works for dynamical systems. First, there is an assumption that prior knowledge about the target domain exists, and without it, most generalization methods would fail [11]. Second, the predominant use of the mean squared error as a loss function is inadequate for evaluating the reconstruction accuracy of chaotic systems. Lastly, the influence of unlabelled trajectory data on the process of learning generalizable dynamical systems remains both unexplored and unresolved — a gap this paper examines for the first time. While switched systems learning methods [19] infer modes by classifying individual data points (analogous to environment inference), our work operates on trajectories governed by ODEs or PDEs.

6 Conclusion

We propose an environment inference method that improves the understanding and generalization of complex dynamical systems across various environments without using manually labeled data. DynaInfer infers environment labels directly from training data, overcoming the challenges associated with explicit annotation. Theoretical analysis ensures convergence of DynaInfer, and experiments show DynaInfer often surpasses non-oracle methods and matches or exceeds oracle performance.

Future research could unfold along the following promising directions: First, to improve generalization in chaotic systems, MSE-based methods could be replaced with more suitable metrics such as the sliced Wasserstein-1 distance, which would require developing a tailored inference model. Similarly, effectively inferring environments for dynamics with complex boundaries remains a significant open problem, as current learning methods often oversimplify boundary conditions. Furthermore, for systems requiring interpretable parameters, our method could be extended to jointly optimize environment labels and physical coefficients. To improve convergence, coordinate optimization techniques may help escape local optima for objectives that are convex in individual variable blocks [13, 30]. Finally, techniques such as adaptive early stopping, dynamic batching, and membership functions [3] could further enhance training efficiency.

Acknowledgments and Disclosure of Funding

This work was supported by NSFC (No. 62425206, 62141607, 62525213), Beijing Municipal Science and Technology Project (No. Z241100004224009), and Big Data and Responsible Artificial Intelligence for National Governance, Renmin University of China. We also extend our thanks to Renzhe Xu and Hao Zou for their insightful discussions.

References

- [1] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.
- [2] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton university press, 2009.
- [3] Federico Bianchi, Alessandro Falsone, Luigi Piroddi, and Maria Prandini. A constrained clustering approach to bounded-error identification of switched and piecewise affine systems. *Automatica*, 146:110589, 2022.

- [4] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.
- [5] Fabio M Carlucci, Antonio D’Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.
- [6] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In Zoubin Ghahramani, editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 193–200. ACM, 2007.
- [7] Emmanuel De Bézenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124009, 2019.
- [8] John R Dormand and Peter J Prince. A family of embedded runge-kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.
- [9] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual review of fluid mechanics*, 51:357–377, 2019.
- [10] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor S. Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17:59:1–59:35, 2016.
- [11] Niclas Göring, Florian Hess, Manuel Brenner, Zahra Monfared, and Daniel Durstewitz. Out-of-domain generalization in dynamical systems reconstruction. In *Proceedings of the 41st International Conference on Machine Learning*, pages 16071–16114, 2024.
- [12] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.
- [13] J William Helton and Orlando Merino. Coordinate optimization for bi-convex matrix inequalities. In *Proceedings of the 36th IEEE Conference on Decision and Control*, volume 4, pages 3609–3613. IEEE, 1997.
- [14] Shoubo Hu, Kun Zhang, Zhitang Chen, and Laiwan Chan. Domain generalization via multidomain discriminant analysis. In *Uncertainty in Artificial Intelligence*, pages 292–302. PMLR, 2020.
- [15] Jing Jiang and ChengXiang Zhai. Instance weighting for domain adaptation in NLP. In John Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics, 2007.
- [16] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.
- [17] Matthieu Kirchmeyer, Yuan Yin, Jérémie Donà, Nicolas Baskiotis, Alain Rakotomamonjy, and Patrick Gallinari. Generalizing to new physical systems via context-informed dynamics model. In *International Conference on Machine Learning*, pages 11283–11301. PMLR, 2022.
- [18] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed Chi. Fairness without demographics through adversarially reweighted learning. *Advances in neural information processing systems*, 33:728–740, 2020.
- [19] Fabien Lauer. Estimating the probability of success of a simple algorithm for switched linear regression. *Nonlinear Analysis: Hybrid Systems*, 8:31–47, 2013.
- [20] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M Hospedales. Episodic training for domain generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1446–1455, 2019.

- [21] Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain generalization with adversarial feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5400–5409, 2018.
- [22] Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*.
- [23] Alfred James Lotka. *Elements of physical biology*. Williams & Wilkins, 1925.
- [24] Gurvan Madec, Romain Bourdallé-Badie, Pierre-Antoine Bouttier, Clément Bricaud, Diego Bruciaferri, Daley Calvert, Jérôme Chanut, Emanuela Clementi, Andrew Coward, Damiano Delrosso, et al. Nemo ocean engine. 2017.
- [25] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 5716–5726. IEEE Computer Society, 2017.
- [26] Aurel Neic, Fernando O Campos, Anton J Prassl, Steven A Niederer, Martin J Bishop, Edward J Vigmond, and Gernot Plank. Efficient computation of electrograms and ecgs in human whole heart simulations using a reaction-eikonal model. *Journal of computational physics*, 346:191–211, 2017.
- [27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [28] John E Pearson. Complex patterns in a simple system. *Science*, 261(5118):189–192, 1993.
- [29] Vihari Piratla, Praneeth Netrapalli, and Sunita Sarawagi. Efficient domain generalization via common-specific low-rank decomposition. In *International Conference on Machine Learning*, pages 7728–7738. PMLR, 2020.
- [30] Tianle Pu, Zijie Geng, Haoyang Liu, Shixuan Liu, Jie Wang, Li Zeng, Chao Chen, and Changjun Fan. RoME: Domain-robust mixture-of-experts for MILP solution prediction across domains. In *Advances in Neural Information Processing Systems*, volume 39, 2025.
- [31] Seonguk Seo, Yumin Suh, Dongwan Kim, Geeho Kim, Jongwoo Han, and Bohyung Han. Learning to optimize domain specific normalization for domain generalization. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pages 68–83. Springer, 2020.
- [32] Sagi Shaier, Maziar Raissi, and Padmanabhan Seshaiyer. Data-driven approaches for predicting spread of infectious diseases through dinns: Disease informed neural networks. *Letters in Biomathematics*, 9(1):71–105, 2022.
- [33] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018.
- [34] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.
- [35] Megha Srivastava, Tatsunori Hashimoto, and Percy Liang. Robustness to spurious correlations via human annotations. In *International Conference on Machine Learning*, pages 9109–9119. PMLR, 2020.
- [36] Baochen Sun and Kate Saenko. Deep CORAL: correlation alignment for deep domain adaptation. In Gang Hua and Hervé Jégou, editors, *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, volume 9915 of *Lecture Notes in Computer Science*, pages 443–450, 2016.

- [37] Wei Sun, Junhui Wang, and Yixin Fang. Regularized k-means clustering of high-dimensional data and its asymptotic consistency. *Electronic Journal of Statistics*, 6:148–167, 2012.
- [38] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.
- [39] Riccardo Volpi, Hongseok Namkoong, Ozan Sener, John C Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. *Advances in neural information processing systems*, 31, 2018.
- [40] Haotian Wang, Kun Kuang, Long Lan, Zige Wang, Wanrong Huang, Fei Wu, and Wenjing Yang. Out-of-distribution generalization with causal feature separation. *IEEE Transactions on Knowledge and Data Engineering*, 36(4):1758–1772, 2023.
- [41] Rui Wang, Robin Walters, and Rose Yu. Meta-learning dynamics forecasting using task inference. *Advances in Neural Information Processing Systems*, 35:21640–21653, 2022.
- [42] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 1(1):1–34, 2020.
- [43] Yuan Yin, Ibrahim Ayed, Emmanuel de Bézenac, Nicolas Baskiotis, and Patrick Gallinari. Leads: Learning dynamical systems that generalize across environments. *Advances in Neural Information Processing Systems*, 34:7561–7573, 2021.
- [44] Weiming Zhi, Tin Lai, Lionel Ott, Edwin V Bonilla, and Fabio Ramos. Learning efficient and robust ordinary differential equations via invertible neural networks. In *International Conference on Machine Learning*, pages 27060–27074. PMLR, 2022.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: The abstract and introduction accurately outline our research questions, and faithfully reflect the paper's contributions and scope.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: We discussed it as future work in the Conclusion section

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: Assumptions, formal statements of theories and proofs can be found in appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: Experiment settings and implementation of methods are described in the experiment section. Codes are available in supplementary materials.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All the datasets in this paper are public with citations (see section 6.1). Code is provided in additional supplemental materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Experiment settings are described in section 4.1 and 4.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We use mean $\pm$ standard deviation to report the results over 5 independent runs. See section 4.2 for details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Refer to Appendix E. Results are averaged over 5 independent runs.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The research conducted in this paper conforms with the NeurIPS Code of Ethics

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: There are no negative social impacts for research conducted in this paper.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The paper poses no safeguards risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and base models in this paper are cited. Codes are credited with original licenses provided.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowd-sourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

A Proof of Proposition 3.1

Proof. Due to the operation in Equation (5), we must have

$$R_{\hat{e}^{(r+1)}}(\theta^{(r+1)}, \phi^{(r+1)}) \leq R_{\hat{e}^{(r+1)}}(\theta^{(r)}, \phi^{(r)}).$$

In addition, according to the operation in Equation (4), we must have

$$\begin{aligned} \forall i \in [N], \int_{t \in I} \left\| \frac{dx_t^i}{dt} - h\left(x_t^i; \theta^{(r)}, \phi_{\hat{e}_i^{(r+1)}}^{(r)}\right) \right\|_2^2 dt \\ \leq \int_{t \in I} \left\| \frac{dx_t^i}{dt} - h\left(x_t^i; \theta^{(r)}, \phi_{\hat{e}_i^{(r)}}^{(r)}\right) \right\|_2^2 dt. \end{aligned}$$

As a result, since the regularization terms (i.e., the $\Omega(\phi_e)$ term) in $R_{\hat{e}^{(r+1)}}(\theta^{(r)}, \phi^{(r)})$ and $R_{\hat{e}^{(r)}}(\theta^{(r)}, \phi^{(r)})$ remain the same, we must have

$$R_{\hat{e}^{(r+1)}}(\theta^{(r)}, \phi^{(r)}) \leq R_{\hat{e}^{(r)}}(\theta^{(r)}, \phi^{(r)}).$$

Now we have

$$R_{\hat{e}^{(r+1)}}(\theta^{(r+1)}, \phi^{(r+1)}) \leq R_{\hat{e}^{(r+1)}}(\theta^{(r)}, \phi^{(r)}) \leq R_{\hat{e}^{(r)}}(\theta^{(r)}, \phi^{(r)}).$$

Now consider the second part of the proposition. Define the following space of θ, ϕ

$$\mathcal{H} \triangleq \left\{ (\theta, \phi) : \exists \hat{e} \in [M]^N \text{ such that } R_{\hat{e}}(\theta, \phi) = \min_{\theta', \phi'} R_{\hat{e}}(\theta', \phi') \right\}.$$

Note that by the assumption, we must have $|\mathcal{H}| < \infty$. Now define $\mathcal{A}$ as

$$\mathcal{A} = \left\{ \int_{t \in I} \left\| \frac{dx_t^i}{dt} - h\left(x_t^i; \theta, \phi_e\right) \right\|_2^2 dt : i \in [N], e \in [M], (\theta, \phi) \in \mathcal{H} \right\}.$$

Note that since $|\mathcal{H}| < \infty$, we have $|\mathcal{A}| < \infty$. C is defined as

$$C = \min_{a, b \in \mathcal{A}, a \neq b} |a - b|.$$

Since $|\mathcal{A}| < \infty$, we must have $C > 0$. In addition, note that when $r > 1$ and $R_{\hat{e}^{(r+1)}}(\theta^{(r+1)}, \phi^{(r+1)}) < R_{\hat{e}^{(r)}}(\theta^{(r)}, \phi^{(r)})$, we must have $\hat{e}^{(r+1)} \neq \hat{e}^{(r)}$. Otherwise we will have

$$\begin{aligned} R_{\hat{e}^{(r+1)}}(\theta^{(r+1)}, \phi^{(r+1)}) &= R_{\hat{e}^{(r)}}(\theta^{(r+1)}, \phi^{(r+1)}) \\ &= R_{\hat{e}^{(r)}}(\theta^{(r)}, \phi^{(r)}). \end{aligned}$$

As a result, there exists $i \in [N]$ such that $\hat{e}_i^{(r+1)} \neq \hat{e}_i^{(r)}$. By the choice of $\hat{e}_i^{(r+1)}$, we must have

$$\begin{aligned} \int_{t \in I} \left\| \frac{dx_t^i}{dt} - h\left(x_t^i; \theta^{(r)}, \phi_{\hat{e}_i^{(r)}}^{(r)}\right) \right\|_2^2 dt \\ \neq \int_{t \in I} \left\| \frac{dx_t^i}{dt} - h\left(x_t^i; \theta^{(r)}, \phi_{\hat{e}_i^{(r+1)}}^{(r)}\right) \right\|_2^2 dt. \end{aligned}$$

As a result, by the definition of C , we have

$$\begin{aligned} \int_{t \in I} \left\| \frac{dx_t^i}{dt} - h\left(x_t^i; \theta^{(r)}, \phi_{\hat{e}_i^{(r)}}^{(r)}\right) \right\|_2^2 dt \\ - \int_{t \in I} \left\| \frac{dx_t^i}{dt} - h\left(x_t^i; \theta^{(r)}, \phi_{\hat{e}_i^{(r+1)}}^{(r)}\right) \right\|_2^2 dt \geq C. \end{aligned}$$

Therefore, we must have

$$R_{\hat{e}^{(r+1)}}(\theta^{(r+1)}, \phi^{(r+1)}) \leq R_{\hat{e}^{(r)}}(\theta^{(r)}, \phi^{(r)}) - C.$$

Now the claim follows. $\square$

B Real-world Experiment

We further evaluated our method using a real-world robot motion trajectory dataset [16]. This dataset comprises three distinct motion patterns: (1) drawing "S" shapes, (2) placing a cube on a shelf, and (3) drawing large "C" shapes. In our experimental framework, each of these patterns is treated as a distinct environment. The objective of this evaluation is to assess whether our method can accurately infer the underlying environment, thereby supporting the learning of a generalizable neural network for simulating their dynamics. To ensure a consistent input structure across environments, all trajectories were projected into a two-dimensional space. The dataset was partitioned into training and test sets with a ratio of 6:1. Furthermore, each trajectory was temporally resampled to a standardized length of 100 time steps for smoothing and uniformity following the practice in [44].

The empirical results are presented in Table 3. In summary, these findings collectively suggest that our approach maintains its efficacy on real-world data, thereby substantiating its practical utility and robustness.

Data	Assignment	LEADS			CoDA- l_1			CoDA- l_2		
		Train MSE	Test MSE	MAPE	Train MSE	Test MSE	MAPE	Train MSE	Test MSE	MAPE
RM	All in One	7.17 E-2	7.41±0.02 E-2	49.22±1.84	7.14 E-2	7.40±0.01 E-2	49.44±3.15	7.17 E-2	7.41±0.00 E-2	39.26±22.13
	One per Env	4.15 E-4	4.91±3.50 E-4	6.68±2.44	8.68 E-4	9.14±0.41 E-4	5.67±1.01	8.18 E-4	8.43±0.39 E-4	5.73±1.19
	Random	7.20 E-2	7.38±0.02 E-2	50.01±1.05	7.12 E-2	7.39±0.01 E-2	48.87±1.81	7.09 E-2	7.39±0.00 E-2	48.86±2.54
	DynaInfer	4.74 E-5	7.93±2.49 E-5	2.83±1.62	9.57 E-5	1.83±3.40 E-4	3.27±2.36	1.71 E-4	1.82±3.07 E-4	2.02±1.66
	Oracle	4.55 E-5	7.02±0.76 E-5	1.78±0.10	1.78 E-5	3.19±0.24 E-5	1.26±0.06	1.99 E-5	2.72±0.18 E-5	1.21±0.08

Table 3: In-domain Experiment Results on the Robot Motion.

C Sensitivity Analysis to $|\mathcal{E}_o|$

To assess the robustness of our method to the number of underlying environments, we conducted a sensitivity analysis using the LV system. We systematically varied the true number of environments, $|\mathcal{E}_o|$, from 2 to 16 and evaluated the performance of our model against the Oracle baseline that has privileged access to the true environment labels. The results are presented in Table 4. DynaInfer, achieves performance comparable to the Oracle across all tested values of $|\mathcal{E}_o|$. This empirical evidence demonstrates that our approach is robust to variations in the quantity of environments.

Table 4: Test Mean Squared Error on the LV system for varying $|\mathcal{E}_o|$

		$\mathcal{E}_o$						
		2	3	4	5	6	7	8
DynaInfer		2.12E-5	4.22E-5	6.77E-5	6.24E-5	8.81E-5	6.70E-5	6.54E-5
Oracle		2.76E-5	3.99E-5	5.75E-5	5.20E-5	7.24E-5	8.14E-5	5.74E-5

		$\mathcal{E}_o$							
		9	10	11	12	13	14	15	16
DynaInfer		6.33E-5	8.34E-5	2.24E-4	1.34E-4	9.69E-5	1.97E-4	3.43E-4	2.73E-4
Oracle		7.60E-5	9.31E-5	1.85E-4	1.23E-4	1.21E-4	1.71E-4	2.56E-4	2.48E-4

D Robustness over M

We assessed our robustness to the overestimation of M , the assumed number of environments, in the LV system. The system's true value is $M^* = 9$, with each environment containing four trajectories. In our experiments, we varied the assumed M from 5 to 36. The results, shown in Table 5, demonstrate that DynaInfer effectively adapts to this overestimation. The learned label count consistently converged to a range of 8–11, which aligns closely with the ground-truth value.

Furthermore, we conducted a comprehensive analysis by sweeping M across its full spectrum, from $M = 1$ to $M = \text{\#trajectory}$. As shown in Table 6, the optimal performance was achieved at $M = 9$ and the model maintained robustness even when M exceeded this value.

Table 5: Learned Label Counts for different values of M averaged over 5 runs

M	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
# Labels	5	6	7	8	9	8.8	9	10.2	9	9	10	10.8	10	11	11	10

M	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
# Labels	10.8	11	9	10	10.2	11	10	10	10	11.2	10	9.2	10	10.8	10	11

Table 6: Test MSE over the full spectrum of M for the LV system

M	MSE	M	MSE	M	MSE	M	MSE
1	7.41E-2	10	2.41E-4	19	2.60E-4	28	2.04E-4
2	4.01E-2	11	3.71E-4	20	1.36E-4	29	3.65E-4
3	2.47E-2	12	2.57E-4	21	1.21E-4	30	1.42E-3
4	1.04E-2	13	1.63E-4	22	2.45E-4	31	3.84E-4
5	7.02E-3	14	2.24E-4	23	9.06E-5	32	1.16E-4
6	4.24E-3	15	1.15E-4	24	2.38E-4	33	3.89E-4
7	2.59E-3	16	1.76E-4	25	8.27E-5	34	3.10E-4
8	1.21E-3	17	1.83E-4	26	1.06E-4	35	1.72E-4
9	7.93E-5	18	3.60E-4	27	3.71E-4	36	2.77E-4

E Environment Specification

We conducted experiments on a server equipped with a 64-core CPU, 256 GB of RAM, and eight 24GB RTX-3090Ti GPUs. The DynaInfer framework was implemented using PyTorch [27]. All NN params in our method are randomly initialized.

Lotka-Volterra (LV) [23] The system models the dynamics between a prey-predator pair in an ecosystem, captured by the following ODE:

$$dm/dt = \alpha m - \beta mn, dn/dt = \delta mn - \gamma n$$

, where m, n represent the population density of the prey and predator, respectively, and $\alpha, \beta, \delta, \gamma$ are the interaction parameters between the two species. The system state is defined as $x_t^e = (m_t^e, n_t^e) \in \mathbb{R}_+^2$ with initial conditions (m_0, n_0) sampled from a uniform distribution $p(x_0) = \text{Unif}([1, 3]^2)$. The environment e is defined by dynamics parameters $\theta_e = (\alpha_e/\beta_e, \gamma_e/\delta_e) \in \Theta$, sampled uniformly from the set Θ . We simulate trajectories over a temporal grid with $\Delta t = 0.5$ and a horizon $T = 10$. At test time, environment labels are inferred from the prediction bias observed over an initial segment of the trajectory of length Δt .

Gray-Scott (GS) [28] The model uses simple reaction-diffusion equations to effectively study complex pattern formation in chemical and biological systems, following underlying PDE dynamics:

$$\begin{aligned}\partial m/\partial t &= D_m \Delta m - mn^2 + F(1 - m), \\ \partial n/\partial t &= D_n \Delta n - mn^2 - (F + k)n.\end{aligned}$$

, where m, n represent the concentrations of two chemical components in the spatial domain S with periodic boundary conditions, and D_m, D_n are their constant diffusion coefficients, and F, k are the reaction parameters that govern the spatio-temporal dynamic patterns. S is a 2D space of dimension 32×32 with spatial resolution of $\Delta s = 2$. The system state $x_t^e = (m_t^e, n_t^e) \in \mathbb{R}_+^{2 \times 32^2}$. We define the initial conditions $(m_0, n_0) \sim p(x_0)$ by uniformly sampling three two-by-two squares, which activate the reactions, from S . $(m_0, n_0) = (1 - \epsilon, \epsilon)$ with $\epsilon = 0.05$ inside the squares and $(m_0, n_0) = (0, 1)$ outside the squares. The environment e is defined by dynamics parameters $\theta = (F_e, k_e) \in \Theta$, sampled uniformly from the environment distribution Q on Θ . We simulate trajectories on a temporal grid using a timestep of $\Delta t = 40$ over a horizon of $T = 400$. At test time, environment labels are inferred from an initial segment of length Δt .

Navier-Stokes (NS) [22] The Navier-Stokes PDE describes the motion of viscous fluid substances:

$$\partial m/\partial t = -n \nabla m + \nu \Delta m + \xi, \nabla v = 0$$

, where n is the velocity field, $m = \nabla \times n$ is the vorticity, both n, m lie in a spatial domain S with periodic boundary conditions, ν is the viscosity (fixed as $1e^{-3}$) and ξ is the constant forcing term in the domain S . The system state is characterized by $x_t^e = m_t^e \in \mathbb{R}^{32^2}$, as initialized per [22]. The environment e is determined by a uniformly sampled forcing term $\xi_e \in \Theta_\xi$. We simulate trajectories across a temporal interval $\Delta t = 1$ over a horizon $T = 10$. At test time, environment labels are inferred from an initial segment of length $2\Delta t$.

The parameters for LV, GS and NS systems are respectively given in Table 7, 8 and 9.

Table 7: Parameters of LV Systems

Params.	Train 1	Train 2	Train 3	Train 4	Train 5	Train 6	Train 7	Train 8	Train 9	Adapt 1	Adapt 2
α	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.7	0.6
β	0.5	0.75	1	0.5	0.5	0.75	0.75	1	1	0.8	0.7
γ	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
δ	0.5	0.5	0.5	0.75	1	0.75	1	0.75	1	0.5	0.5

Table 8: Parameters of GS Systems

Params.	Train 1	Train 2	Train 3	Adapt 1	Adapt 2
F	0.037	0.03	0.039	0.033	0.036
k	0.06	0.062	0.058	0.059	0.061

Table 9: Parameters of NS Systems

	ξ
Train 1	$0.1 * (\sin(2\pi(X + Y)) + \cos(2\pi * (X + Y)))$
Train 2	$0.1 * (\sin(2\pi(X + Y)) + \cos(2\pi * (X + 2Y)))$
Train 3	$0.1 * (\sin(2\pi(X + Y)) + \cos(2\pi * (2X + Y)))$
Train 4	$0.1 * (\sin(2\pi(X + 2Y)) + \cos(2\pi * (2X + Y)))$
Adapt 1	$0.1 * (\sin(2\pi(2X + Y)) + \cos(2\pi * (X + 2Y)))$
Adapt 2	$0.1 * (\sin(2\pi(2X + Y)) + \cos(2\pi * (2X + Y)))$

F Centroid-like NN Behavior

We conducted an additional experiment measuring the MSE loss for each training sample across different neural networks (NNs). The results, shown in Table 10, yielded two key findings: first, each training sample has a uniquely best-fit NN where it attains minimal loss. Second, this optimal NN generalizes effectively to a broader cluster of trajectories—specifically, those originating from the same environment. This alignment between a "centroid-like" NN and environment-specific trajectory clusters strongly reinforces our methodological rationale.

Traj.	NN1	NN2	NN3	NN4	NN5	NN6	NN7	NN8	NN9	Assign.
1	2.06e-01	1.18e-01	2.80e-01	7.76e-06	8.34e-02	1.59e-01	3.14e-01	2.50e-01	2.35e-01	3
2	2.05e-01	1.21e-01	2.66e-01	9.35e-06	8.42e-02	1.53e-01	3.36e-01	2.40e-01	2.26e-01	3
3	2.05e-01	1.21e-01	2.65e-01	1.03e-05	8.41e-02	1.52e-01	3.35e-01	2.39e-01	2.25e-01	3
4	2.45e-01	1.51e-01	3.85e-01	1.98e-05	1.13e-01	1.89e-01	4.05e-01	3.10e-01	3.12e-01	3
5	3.31e-02	1.65e-01	1.12e-01	8.32e-02	5.36e-06	5.42e-02	3.68e-01	7.26e-02	1.37e-01	4
6	3.15e-02	1.87e-01	1.07e-01	8.46e-02	4.26e-06	5.57e-02	4.21e-01	6.80e-02	1.44e-01	4
7	3.14e-02	1.86e-01	1.06e-01	8.46e-02	4.50e-06	5.52e-02	4.19e-01	6.74e-02	1.43e-01	4
8	2.84e-02	2.68e-01	2.13e-01	1.14e-01	2.22e-05	9.57e-02	5.38e-01	1.18e-01	2.36e-01	4
9	7.60e-06	2.32e-01	5.48e-02	2.07e-01	3.36e-02	4.44e-02	4.22e-01	1.91e-02	1.18e-01	0
10	6.45e-06	2.64e-01	6.07e-02	2.05e-01	3.12e-02	5.17e-02	4.93e-01	2.10e-02	1.37e-01	0
11	6.65e-06	2.63e-01	6.02e-02	2.05e-01	3.11e-02	5.14e-02	4.91e-01	2.07e-02	1.36e-01	0
12	1.95e-05	3.66e-01	1.61e-01	2.45e-01	2.92e-02	9.83e-02	6.31e-01	7.05e-02	2.32e-01	0
13	2.31e-01	1.35e-05	1.56e-01	1.16e-01	1.64e-01	8.52e-02	5.28e-02	1.84e-01	7.04e-02	1
14	2.63e-01	1.39e-05	1.64e-01	1.22e-01	1.87e-01	9.48e-02	5.95e-02	2.02e-01	7.16e-02	1
15	2.62e-01	1.41e-05	1.63e-01	1.22e-01	1.86e-01	9.44e-02	5.93e-02	2.01e-01	7.13e-02	1
16	3.65e-01	1.89e-05	2.33e-01	1.50e-01	2.67e-01	1.19e-01	6.96e-02	2.56e-01	1.08e-01	1
17	4.22e-01	5.35e-02	2.43e-01	3.15e-01	3.68e-01	1.95e-01	2.78e-05	3.17e-01	1.13e-01	6
18	4.91e-01	5.91e-02	2.77e-01	3.35e-01	4.19e-01	2.27e-01	2.12e-05	3.66e-01	1.30e-01	6
19	4.89e-01	5.89e-02	2.76e-01	3.34e-01	4.17e-01	2.26e-01	2.08e-05	3.64e-01	1.30e-01	6
20	6.28e-01	6.90e-02	3.26e-01	4.04e-01	5.35e-01	2.56e-01	1.99e-05	4.18e-01	1.52e-01	6
21	4.36e-02	8.60e-02	1.90e-02	1.58e-01	5.31e-02	7.09e-06	1.96e-01	1.98e-02	2.19e-02	5
22	5.16e-02	9.53e-02	1.74e-02	1.54e-01	5.58e-02	8.08e-06	2.28e-01	2.09e-02	2.34e-02	5
23	5.13e-02	9.49e-02	1.73e-02	1.53e-01	5.53e-02	7.86e-06	2.27e-01	2.08e-02	2.33e-02	5
24	9.81e-02	1.19e-01	3.84e-02	1.89e-01	9.61e-02	1.45e-05	2.56e-01	2.72e-02	3.41e-02	5
25	1.17e-01	7.14e-02	2.46e-02	2.35e-01	1.36e-01	2.11e-02	1.15e-01	5.47e-02	8.69e-06	8
26	1.37e-01	7.21e-02	2.82e-02	2.26e-01	1.44e-01	2.36e-02	1.31e-01	6.42e-02	5.20e-06	8
27	1.36e-01	7.18e-02	2.81e-02	2.26e-01	1.43e-01	2.35e-02	1.31e-01	6.39e-02	5.32e-06	8
28	2.32e-01	1.08e-01	3.35e-02	3.12e-01	2.37e-01	3.47e-02	1.52e-01	7.46e-02	8.12e-06	8
29	1.90e-02	1.84e-01	1.02e-02	2.49e-01	7.19e-02	2.00e-02	3.17e-01	2.50e-06	5.57e-02	7
30	2.07e-02	2.03e-01	1.15e-02	2.41e-01	6.79e-02	2.11e-02	3.68e-01	2.91e-06	6.44e-02	7
31	2.04e-02	2.02e-01	1.14e-02	2.40e-01	6.74e-02	2.10e-02	3.66e-01	3.05e-06	6.41e-02	7
32	7.07e-02	2.55e-01	2.04e-02	3.09e-01	1.20e-01	2.74e-02	4.17e-01	8.74e-06	7.41e-02	7
33	5.48e-02	1.56e-01	1.22e-06	2.80e-01	1.11e-01	1.89e-02	2.44e-01	1.00e-02	2.53e-02	2
34	6.04e-02	1.65e-01	3.09e-06	2.67e-01	1.07e-01	1.76e-02	2.79e-01	1.14e-02	2.83e-02	2
35	5.99e-02	1.64e-01	3.10e-06	2.66e-01	1.06e-01	1.75e-02	2.78e-01	1.14e-02	2.82e-02	2
36	1.61e-01	2.32e-01	1.20e-05	3.85e-01	2.14e-01	3.90e-02	3.26e-01	2.06e-02	3.35e-02	2

Table 10: Training MSE for Trajectories by each Neural Network and Label Assignment

G Environment Assignment Convergence on GS

Figure 6 presents the temporal evolution of environment assignment probabilities using the LEADS base model on GS.

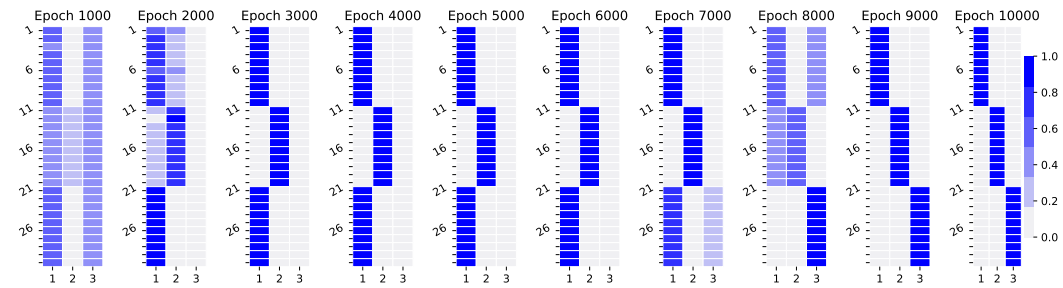


Figure 6: Environment Assignment Probability over Time with LEADS as Base Model on GS. Despite initial inaccuracies due to complex dynamics, the assignment ultimately converges to the correct label.

H Plots on Learned Dynamics

We present the recovered test trajectories produced by the learned neural network. Figures 7, 8, and 9 illustrate GS; Figures 10, 11, and 12 depict NS; and Figure 13 shows LV. A close examination reveals that the trajectories predicted by DynaInfer closely align with both the Oracle and the ground truth for the selected systems.

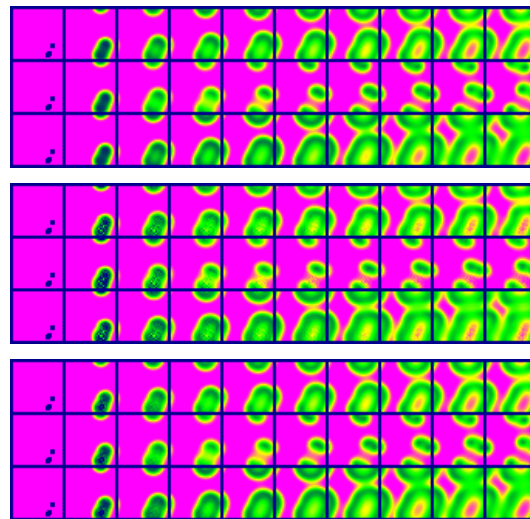


Figure 7: Comparison of final GS states predicted by DynaInfer (bottom) and Oracle (middle) against the ground truth (top) with base model LEADS.

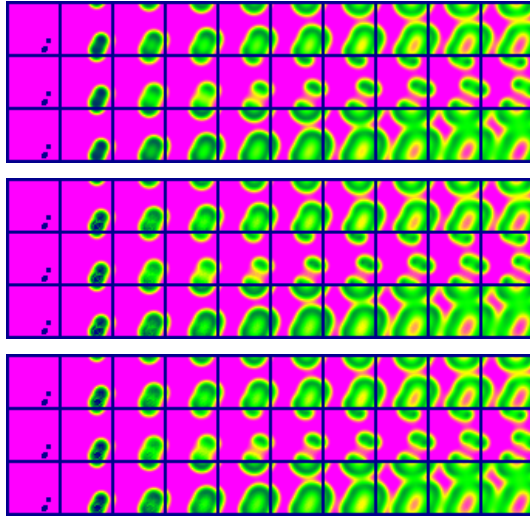


Figure 8: Comparison of final GS states predicted by DynaInfer (bottom) and Oracle (middle) against the ground truth (top) with base model CoDA- l_1 .

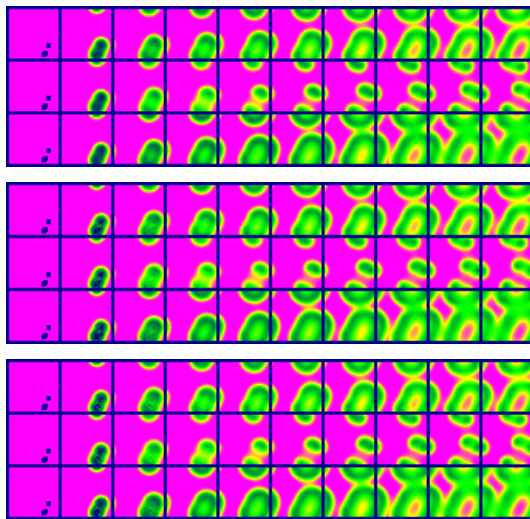


Figure 9: Comparison of final GS states predicted by DynaInfer (bottom) and Oracle (middle) against the ground truth (top) with base model CoDA- l_2 .

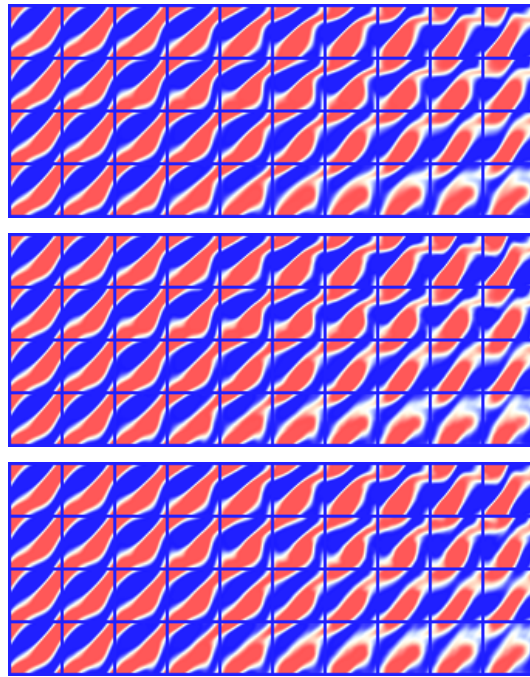


Figure 10: Comparison of final NS states predicted by DynaInfer (bottom) and Oracle (middle) against the ground truth (top) with base model LEADS.

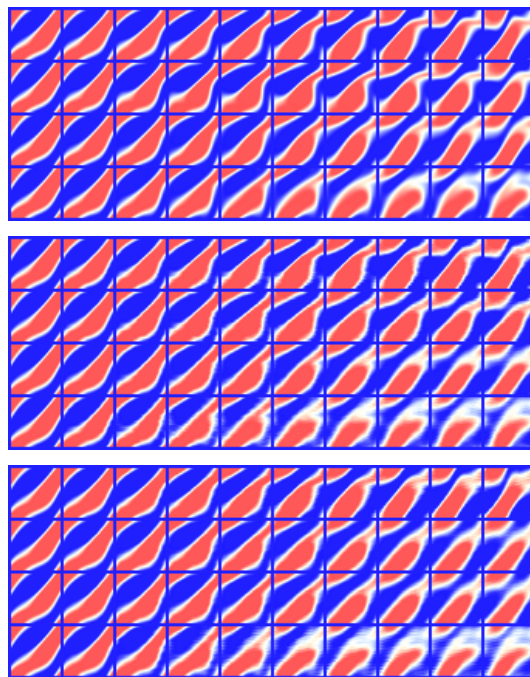


Figure 11: Comparison of final NS states predicted by DynaInfer (bottom) and Oracle (middle) against the ground truth (top) with base model CoDA- l_1 .

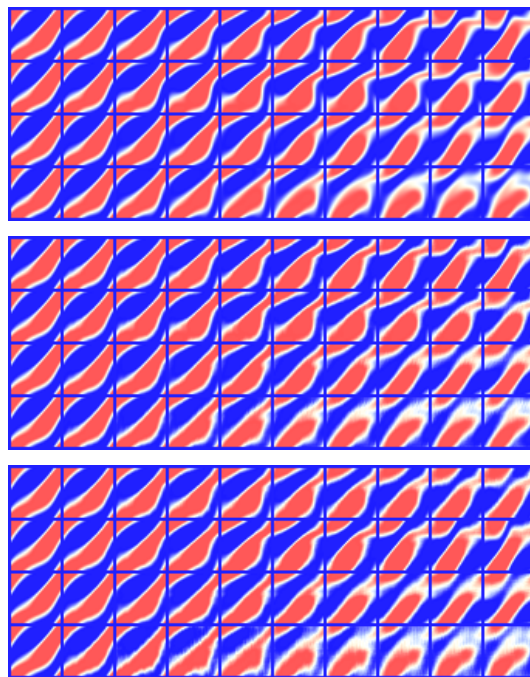


Figure 12: Comparison of final NS states predicted by DynaInfer (bottom) and Oracle (middle) against the ground truth (top) with base model CoDA- l_2 .

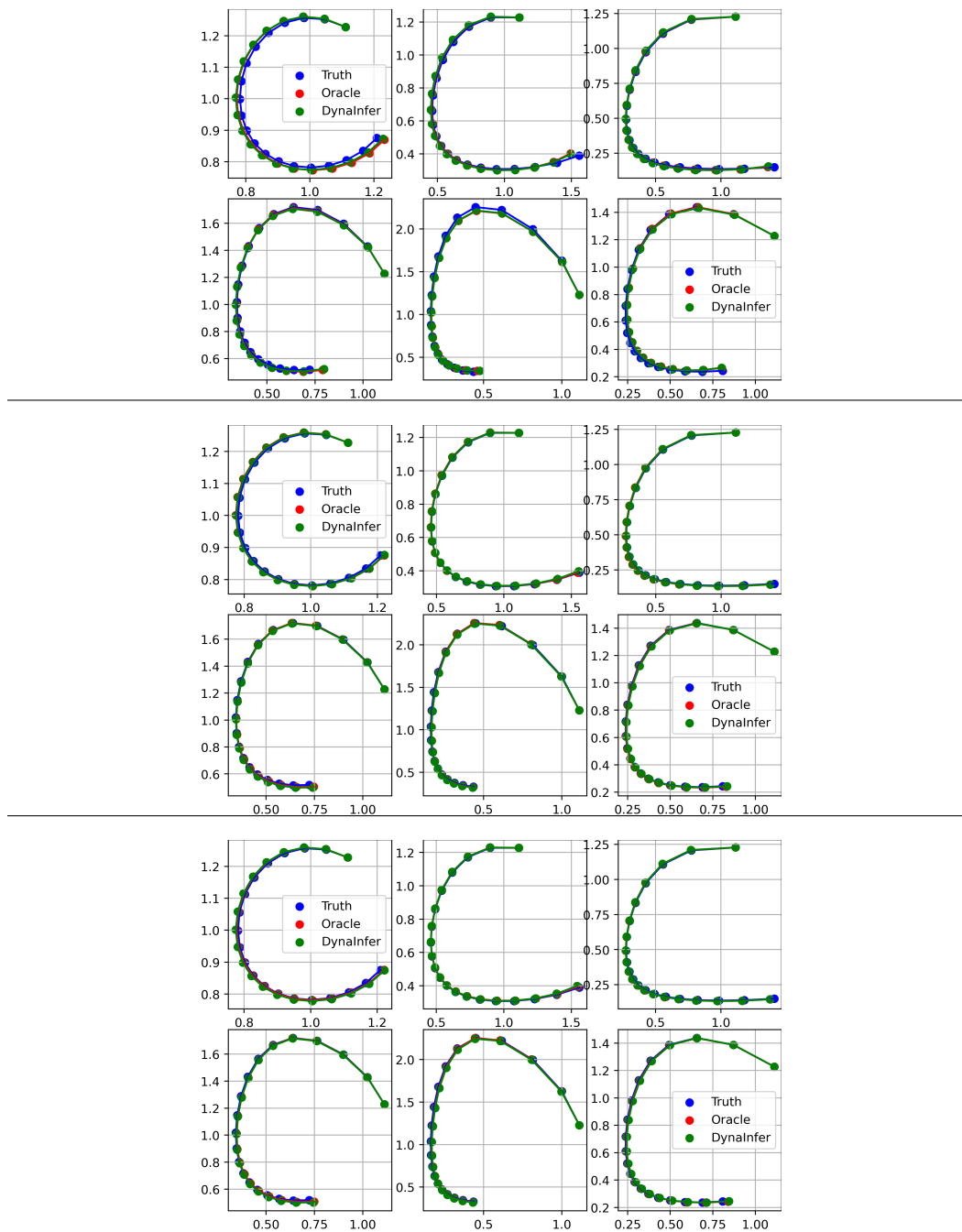


Figure 13: Comparison of predicted LV trajectories from 6 environments by DynaInfer (green) and Oracle (red) against the ground truth (blue) with base models LEADS (top), CoDA- l_1 (middle), and CoDA- l_2 (bottom).