DynaInfer: Environment Inference for Learning Generalizable Dynamical System

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Data-driven methods, especially neural network-based emulators, offer cost-effective, robust alternatives to conventional approaches for analyzing complex dynamic systems across various domains. Despite these advancements, there is a continued reliance on the assumption that data are independent and identically distributed, which has driven the development of generalization techniques that handle environmental differences to improve performance. However, these methods frequently encounter limitations due to their dependence on environment labels, which are often unavailable during training because of data acquisition challenges, privacy constraints, and environmental variability, particularly in large public datasets and domains with strict privacy regulations. In response, we introduce DynaInfer, an innovative method that infer environment specification by clustering the prediction errors with fixed neural networks at current train round, thus revealing useful environment assignments directly from the training data. We prove that our algorithm could effectively solve the alternate optimization problem under unlabeled scenarios and conduct extensive experiments across diverse dynamic systems, representing multiple application domains. Experiment results demonstrate that our method not only outperforms other environment assignment techniques but also rapidly converges to the true labels. Interestingly, this approach could yield superior outcomes even when environment labels are readily available.

1 Introduction

Data-driven approaches, particularly neural networks, provide a robust alternative or enhancement to traditional physics-based methodologies for comprehending complex dynamic systems [3]. Neural network-based emulators are valuable for their rapid, budget-friendly approximations of intricate simulations [8, 18], especially advantageous when physics are poorly understood or misinterpreted, or modeling of external disturbances is lacking [37, 29]. These emulators excel at managing large variable sets and resolving issues challenging for conventional solvers. With recent advances in deep learning and novel methods for modeling temporal and spatio-temporal systems, there is a surge in contributions across fields from simple Hamiltonian dynamics to complex areas like fluid dynamics and climatology [27, 6].

While recent advancements have yielded promising results, they often intrinsically presuppose ideal conditions of abundant, static data to validate the independent and identically distributed (i.i.d.) hypothesis. It has been noted in recent literature, however, that this hypothesis can be easily violated in light of the practical challenges and expenses associated with data collection, as well as the potential evolution of the environment due to some exogenous factors [20, 22]. Recent efforts in modeling dynamical systems have introduced generalization methods that account for variations across different environments, enabling them to avoid the pitfall of learning an averaged model that frequently underperforms [37, 14].

Nevertheless, a significant limitation often encountered with these generalization techniques is their inherent requirement for partitioning datasets across multiple domains or environments. These environment assignments should implicitly implicitly characterize their distinct variations, enabling generalization algorithms to identify and leverage both similarities and discrepancies. However, it is common that these essential environmental labels are not accessible during the training phase, primarily due to the challenges in data acquisition or privacy restrictions. For instance, in scientific research, there may be circumstances where data gets collected over time and under varying conditions that are not entirely controlled or known [36]. Specifically, in ecological studies, certain parameters specifying the environments, such as temperature, rainfall, or other natural phenomena may vary and might not be comprehensively recorded or known [2]. Moreover, when data is aggregated from multiple sources or databases, environment labels risk being lost or unrecorded, a situation that is particularly common in expansive public datasets [30]. Lastly, in fields where data privacy is of paramount importance – such as healthcare, banking, or social networking – access to information about the originating environment of the data is often tightly controlled [15].

In response to the unknown environment label challenge, our aim is to avoid manual environment specification via an innovative approach, inspired by k-means clustering. The core premise of our approach is that trajectories within the same environment exhibit consistent dynamics and similar prediction losses when predicted using a neural network. This similarity enables us to identify meaningful environment assignments directly from the training data. Further, we introduce an environment inference objective for dynamic system that is designed to minimize environment-specific loss. Upon inferring environments using fixed neural networks, we then update these networks with the inferred environments, aiming to learn a generalizable dynamic system.

Our developed method, DynaInfer, is designed to identify environment labels from mixed dynamic system trajectories, enabling the training of off-the-shelf generalization algorithms for dynamic systems in settings where such labels are unavailable. Notably, our results show that inferring environments directly from mixed sequence data can enhance the effectiveness of generalization strategies, even when manual environment assignment is available.

Our main contributions are as follows:

- We are the first to explore the challenge of unknown environment labels for dynamic systems and propose a general framework named DynaInfer that utilizes the prediction loss to accurately infer environments from mixed sequence data.
- We prove that our algorithm is capable of effectively addressing the challenge of alternate optimization problems in the absence of labeled data.
- We demonstrate the efficacy of DynaInfer through experiments in both in-domain settings and adaptation scenarios using three representative dynamic systems. Results confirm that the environment labels assigned by DynaInfer converge rapidly to the true labels.

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,

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

where $t \in \mathbb{R}$ is the time index within a set time interval $I = [0, T]$, and x_t is a time-variant state within a bounded set $\mathcal{A}$. The evolution function $f : \mathcal{A} \to T\mathcal{A}$ maps x_t to its temporal derivatives in the tangent space $T\mathcal{A}$ and is a component of the 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 characterie 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 Dynamic Systems Learning

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

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 function shared across all environments, parameterized by θ, and an environment-specific function, parameterized by ϕ_e for each environment e. The set of environment-specific parameters is denoted by $\boldsymbol{\phi} = \{\phi_e\}_{e \in [M]}$. Consequently, the dynamics of each trajectory x^i are parameterized by both the universal and environment-specific parameters,

$$\frac{dx_t^i}{dt} = h\left(x_t^i; \theta, \phi_e\right).$$

This parametrization represents a decomposition at the functional level ($h\left(x_t^i; \theta, \phi_e\right) = f_\theta(x_t^i) + g_{\phi_e}(x_t^i)$, as in [37]) or at the parametric level ($h\left(x_t^i; \theta, \phi_e\right) = f_{\theta+\phi_e}(x_t^i)$, as in [14]). 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 $\boldsymbol{e} = \{e_1, e_2, \ldots, e_N\}$, the loss function is given by,

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

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 $\Omega(\phi_e)$ serves a regularization term for ϕ_e, with λ controls 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 $\boldsymbol{e}$. Specifically, our aim is to learn the environment assignment $\hat{\boldsymbol{e}} = \{\hat{e}^1, \hat{e}^2, \ldots, \hat{e}^n\} \in [M]^N$ for each trajectory, to optimize Equation (2) effectively. The overall objective is defined as follows:

$$\hat{\boldsymbol{e}}^*, \theta^*, \boldsymbol{\phi}^* = \underset{\hat{\boldsymbol{e}}, \theta, \boldsymbol{\phi}}{\arg\min}\, R_{\hat{\boldsymbol{e}}}(\theta, \boldsymbol{\phi}). \tag{3}$$

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 realistic model that maintains favourable performance even when the exact count of true environments is unknown.

3 The DynaInfer Framework

In this section, we introduce a novel environment inference framework that operates without prior domain knowledge, proving especially effective in dynamic systems where exogenous factors are unobserved and in situations where relevant environmental information is unclear or absent.

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, $\boldsymbol{\phi}$. The first step in our approach centers on inferring environment labels by analyzing the prediction errors in the trajectories as 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 the refinement of the neural network parameters based on the newly inferred environment assignments in an unbiased way, enabling the neural network to precisely adapt to the uniqueness 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 1.

Algorithm 1 DynaInfer framework

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

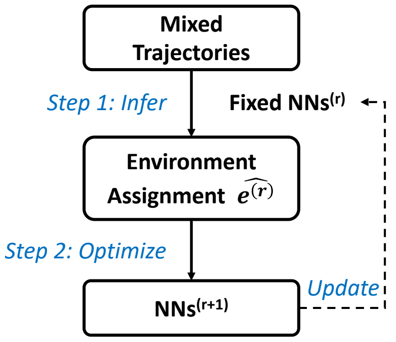

Figure 1: Model Framework.

3.1 Bias-informed 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 careful speculation, we observe that the optimization objective presented in Equation (2) exhibits conceptual similarities to the one of k-means clustering. In k-means clustering, the primary goal is to minimize the within-cluster sum of squares, often referred to as inertia or cluster inertia [32]. 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 the "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—comparable to initializing a random centroid in k-means—we assign each trajectory a label based on the minimal prediction loss calculated from all available networks. Subsequently, we "refine this centroid" through optimization specifically aimed at minimizing the loss as defined in Equation (2). With this iterative optimization process, we can achieve the objective described in Equation (3).

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

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

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 as stated in Proposition 3.1.

3.2 Assignment-directed 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 $\theta^{(r)}, \boldsymbol{\phi}^{(r)}$, is given by

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

3.3 Theoretical Property

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

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

$$R_{\hat{\mathbf{e}}^{(r+1)}}\left(\theta^{(r+1)}, \boldsymbol{\phi}^{(r+1)}\right) \le R_{\hat{\mathbf{e}}^{(r)}}\left(\theta^{(r)}, \boldsymbol{\phi}^{(r)}\right).$$

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

$$R_{\hat{\mathbf{e}}^{(r+1)}}\left(\theta^{(r+1)}, \boldsymbol{\phi}^{(r+1)}\right) \le R_{\hat{\mathbf{e}}^{(r)}}\left(\theta^{(r)}, \boldsymbol{\phi}^{(r)}\right) - C.$$

Remark 3.1. Given the assumptions made in prior works [37, 14] 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 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 explore three distinct dynamical systems, each governed by a specific type of differential equation: an ODE for biological modeling, PDEs for reaction-diffusion processes in chemistry, and the Navier-Stokes equations for incompressible Newtonian fluid dynamics. These systems, characterized by complex and nonlinear dynamics, challenge our method's ability to classify spatiotemporal patterns and physical laws across diverse environments.

4.1 Environment Specification

Lotka-Volterra (LV) [19] 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 $p(x_0) = Unif([1, 3]^2)$. 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 $T = 10$.

Gray-Scott (GS) [24] The model uses simple reaction-diffusion equations to effectively study complex pattern formation in chemical and biological systems, following underlying PDE dynamics:

$$\partial m/\partial t = D_m \Delta m - mn^2 + F(1-m), \partial n/\partial t = D_n \Delta n - mn^2 - (F+k)n$$

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 $T = 400$.

Navier-Stokes (NS) [18] 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 [18]. 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 $T = 10$.

4.2 Experimental Setting and Baselines

Settings We validate DynaInfer across two 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 inferred based on the bias between predicted and true trajectories over the first temporal interval Δt. For adaptation experiments, after initial training on $\mathcal{E}_o$, we fine-tune and test on $\mathcal{E}_u$ where environment labels are known. We report the mean and standard deviation of the Mean Squared Error (MSE) across test trajectories, averaged over five independent runs.

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 B. 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 [7, 11], while the NS data is simulated with the pseudo-spectral method as in [18].

Baselines We explore three potential strategies for assigning environment labels in the absence of environmental information against us (DynaInfer): grouping all samples into a single environment (All in One), offering each sample a distinct environment label (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 dynamic system generalization: LEADS [37], CoDA-l_1, and CoDA-l_2 [14]. 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.

4.3 Experimental Results

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

Data	Assignment	LEADS		CoDA-l_1		CoDA-l_2	
		Train	Test	Train	Test	Train	Test
LV	All in One	7.17 E-2	7.41±0.02 E-2	7.14 E-2	7.40±0.01 E-2	7.17 E-2	7.41±0.00 E-2
	One per Env	4.15 E-4	4.91±3.50 E-4	8.68 E-4	9.14±0.41 E-4	8.18 E-4	8.43±0.39 E-4
	Random	7.20 E-2	7.38±0.02 E-2	7.12 E-2	7.39±0.01 E-2	7.09 E-2	7.39±0.00 E-2
	DynaInfer	**4.74 E-5**	**7.93±2.49 E-5**	**9.57 E-5**	**1.83±3.40 E-4**	**1.71 E-4**	**1.82±3.07 E-4**
	Oracle	4.55 E-5	7.02±0.76 E-5	1.78 E-5	3.19±0.24 E-5	1.99 E-5	2.72±0.18 E-5
GS	All in One	8.73 E-3	9.60±0.02 E-3	9.24 E-3	9.61±0.03 E-3	9.25 E-3	9.60±0.00 E-3
	One per Env	1.38 E-3	1.65±0.54 E-3	1.56 E-3	1.91±0.06 E-3	1.52 E-3	1.87±0.02 E-3
	Random	8.78 E-3	9.36±0.20 E-3	9.25 E-3	9.59±0.03 E-3	8.77 E-3	9.35±0.02 E-3
	DynaInfer	**3.60 E-5**	**4.14±0.21 E-5**	**9.22 E-5**	**1.23±0.41 E-4**	**6.69 E-5**	**7.25±2.11 E-5**
	Oracle	7.73 E-5	1.34±0.76 E-4	6.04 E-5	9.60±3.91 E-5	4.69 E-5	7.04±1.84 E-5
NS	All in One	5.34E-02	6.71±0.11 E-2	5.79E-02	6.64±0.11 E-2	6.17E-02	6.64±0.03 E-2
	One per Env	2.24E-02	4.11±0.14 E-2	3.45E-02	3.88±0.22 E-2	2.31E-02	4.04±0.22 E-2
	Random	3.06E-02	6.58±0.05 E-2	5.04E-02	6.58±0.05 E-2	5.78E-02	6.66±0.04 E-2
	DynaInfer	**6.10E-04**	**7.05±0.34 E-3**	**1.23E-02**	**1.62±0.18 E-2**	**8.92E-04**	**1.19±0.12 E-2**
	Oracle	2.59E-04	6.55±1.34 E-3	1.36E-02	1.73±0.29 E-2	7.11E-04	9.46±0.51 E-3

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. The "One per Env" strategy, although only yielding mediocre results, provides a feasible initial approach for startups. In all dataset, DynaInfer significantly outperforms these assignment strategies, demonstrating evident improvements in performance. Furthermore, DynaInfer displays consistent effectiveness across all tested base models and datasets, highlighting its robustness against varying generalization methods and datasets. Notably, DynaInfer either approaches or surpasses Oracle (especially in complex PDE environments such as GS and NS), suggesting that its bias-informed method might compensate effectively for not having access to 'cheating labels' presumed by Oracle. In Appendix C, we demonstrate that the final states obtained with DynaInfer qualitatively align closely with the ground truth and the Oracle.

Table 2: Adaption Experiment Results on the LV, GS, and NS environment. 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.

Data	Assignment	All in One	One per Env	Random	DynaInfer	Oracle
LV	LEADS	4.16±8.61 E-2	2.28±1.81 E-3	1.72±0.53 E-3	**5.77±1.46 E-4**	1.67±2.26 E-3
	CoDA-l_1	4.01±6.43 E-2	1.72±0.53 E-3	1.14±0.61 E-3	**8.37±0.94 E-5**	5.85±1.24 E-5
	CoDA-l_2	4.10±7.61 E-2	1.66±0.82 E-3	1.05±0.54 E-3	**8.49±2.07 E-5**	5.12±3.17 E-5
GS	LEADS	4.59±1.18 E-4	4.53±2.17 E-3	5.73±0.86 E-4	**1.00±0.32 E-4**	2.21±0.93 E-4
	CoDA-l_1	2.85±0.55 E-3	1.08±0.82 E-3	2.92±0.80 E-3	**2.41±0.91 E-4**	2.66±0.79 E-4
	CoDA-l_2	2.76±0.72 E-3	1.19±0.97 E-3	2.84±0.89 E-3	**2.13±0.41 E-4**	2.10±0.89 E-4
NS	LEADS	4.01±6.51 E-2	3.78±2.08 E-2	1.32±0.53 E-2	**7.52±0.76 E-3**	1.16±0.68 E-2
	CoDA-l_1	1.25±0.20 E-2	2.04±0.68 E-2	4.66±1.04 E-2	**9.27±1.81 E-3**	7.46±0.72 E-3
	CoDA-l_2	1.29±0.29 E-2	2.43±0.48 E-2	4.37±0.99 E-2	**9.71±2.10 E-3**	7.32±0.81 E-3

Adaptation Results The adaptation results displayed in Table 2 highlight various assignment strategies' performance. The "One per Env" strategy typically outperforms the "All in One" approach. Although the "Random" assignment benefits the base model LEADS, it slightly detracts from the performance of CoDA across all three datasets. DynaInfer exhibits favourable adaptation capabilities across different datasets and base generalization methods, consistently surpassing other non-Oracle techniques. This performance suggests that DynaInfer effectively learns commonalities among environments, facilitating easier adaptation to new conditions. Moreover, the performance disparity between DynaInfer and the Oracle is noticeably smaller in these adaptation experiments than in in-domain generalization.

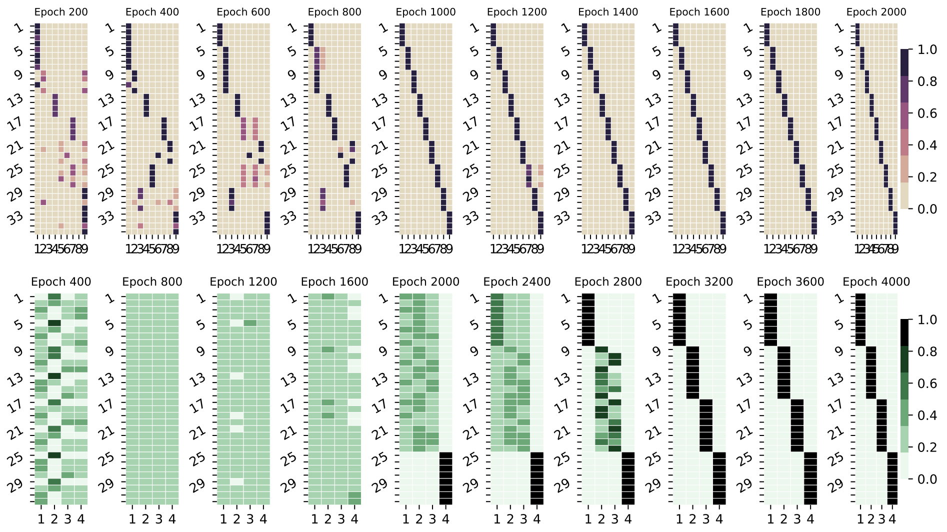

Figure 2: Environment assignment probability over time, averaged over 5 runs, with LEADS as base model (on LV (top) and NS (bottom); see Appendix for GS). The assignment converges to the true label faster than the designated training step. A similar trend is observed with the CoDA model.

Assignments Convergence The probability of environment assignments over training time is illustrated in Figure 5. Initially, our model could render random assignments because the neural networks are not yet fully optimized. However, the assignments quickly converge to the true labels. We observe that dynamic systems with simpler dynamics, such as LV compared to NS, facilitate quicker learning of base generalization methods, leading to faster convergence of environment assignments.

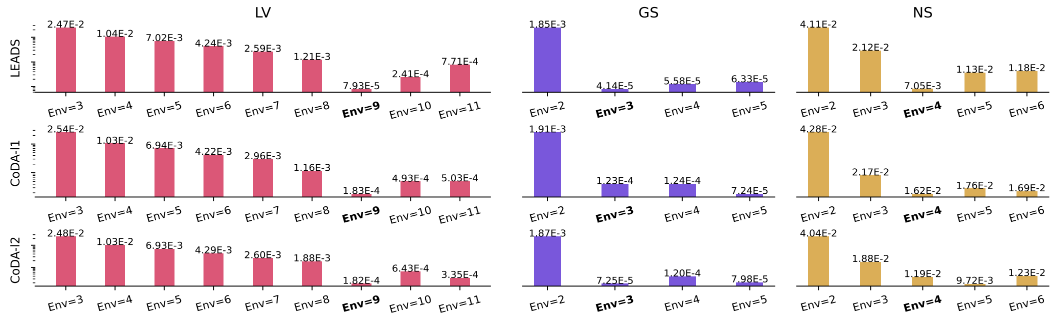

Figure 3: Performance across different assumed environments with DynaInfer. The y-axis is displayed on a logarithmic scale, and the true number of environments is marked in bold on the x-axis. The peak performance aligns with the true number of environments with high probability, and remains stable thereafter.

Examining Performance across Assumed Environments As the true number of environments ($|\mathcal{E}_o|$) may be unknown, we investigate how the performance of DynaInfer varies with different assumed number of environments M, as shown in Figure 5. Our results highlight the utility of prior knowledge about the true M: performance typically peaks when the assumed M matches the true value. Additionally, we find that our model demonstrates relative insensitivity to over-estimations of M. It appears capable of effectively clustering trajectories from the same environment by accounting for bias, without significant disruption from an excess of inaccurately trained neural networks when the assumed M is overly large. Moreover, even when M is underestimated, our method consistently outperforms other non-oracle approaches across most values of M.

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 [17, 12, 25, 26]. Other approaches enhance the diversity of training data by augmenting source domains [4, 28, 34]. Additionally, some methods leverage meta-learning and invariant risk minimization for regularization, further enhancing generalization [16, 1].

Domain adaptation (DA) methods enable model generalization to target domains with shifted data and are primarily classified into three categories. Instance-based methods reweight or adjust training samples to reflect the test distribution [13, 5]. Feature-based approaches align features across training and test distributions [33, 31]. Model-based strategies focus on developing models that are either robust to domain shifts or specifically tailored for the test domain [21, 9].

5.2 Generalization for Dynamical Systems

Generalization in dynamical systems remains underexplored in literature. Among limited studies, the LEADS strategy emerges as a novel multi-task learning framework, effectively generalizing across the functional space of dynamic systems [37]. 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 [14]. Conversely, DyAd, a context-aware meta-learning approach, adjusts the dynamics model by decoding a time-invariant context from observed states [35]. Despite its novelty, DyAd relies on possibly impractical weak supervision based on physics-derived quantities and uses Adaptive Instance Normalization, a type of hypernetwork decoding that could degrade performance.

Currently, for generalization works in dynamic systems, three notable weaknesses prevail. First, there is an assumption that prior knowledge about the target domain exists, and without it, most generalization methods would fail [10]. 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 dynamic systems remains both unexplored and unresolved — a gap this paper examines for the first time.

6 Conclusion

In this paper, we proposed an environment inference method to enhance the understanding and generalization of complex dynamical systems across various environments without relying on manually labeled environment data. DynaInfer successfully infers environment labels from training data, circumventing the traditional challenges associated with explicit environment annotation. Our theoretical analysis guarantee convergence for DynaInfer, and extensive experimental results across a diverse set of nonlinear dynamics demonstrate that DynaInfer not only surpasses non-oracle approaches, but also matches but occasionally surpasses oracle.

Future research could unfold in at least three promising directions. First, our current work is limited by its reliance on MSE-based generalization methods, which may underperform in chaotic systems [10]. Adopting alternative regression error metrics, such as the sliced Wasserstein-1 distance, could improve generalization in these contexts, necessitating the development of a tailored environment inference model for chaotic systems. Second, exploring methodologies with a focus on fairness and devising innovative label assignment strategies presents another fertile research avenue. Lastly, integrating heuristics and Bayesian methods into error evaluation processes has the potential to significantly enhance efficiency, providing another exciting direction for further study.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. Turbulence modeling in the age of data. *Annual review of fluid mechanics*, 51:357–377, 2019.

[9] 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.

[10] Niclas Göring, Florian Hess, Manuel Brenner, Zahra Monfared, and Daniel Durstewitz. Out-of-domain generalization in dynamical systems reconstruction. *arXiv preprint arXiv:2402.18377*, 2024.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[19] Alfred James Lotka. *Elements of physical biology*. Williams & Wilkins, 1925.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] John E Pearson. Complex patterns in a simple system. *Science*, 261(5118):189–192, 1993.

[25] 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.

[26] 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.

[27] Sagi Shaier, Maziar Raissi, and Padmanabhan Seshaiyer. Data-driven approaches for predicting spread of infectious diseases through dinns: Disease informed neural networks. *arXiv preprint arXiv:2110.05445*, 2021.

[28] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745*, 2018.

[29] Justin Sirignano and Konstantinos Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018.

[30] 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.

[31] 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.

[32] Wei Sun, Junhui Wang, and Yixin Fang. Regularized k-means clustering of high-dimensional data and its asymptotic consistency. 2012.

[33] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *CoRR*, abs/1412.3474, 2014.

[34] 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.

[35] Rui Wang, Robin Walters, and Rose Yu. Meta-learning dynamics forecasting using task inference. *Advances in Neural Information Processing Systems*, 35:21640–21653, 2022.

[36] 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.

[37] 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.

A Proof of Proposition 3.1

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

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

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

$$\forall i \in [N], \int_{t\in I} \left\| \frac{dx_t^i}{dt} - h\left(x_t^i; \theta^{(r)}, \phi^{(r)}_{e_i^{(r+1)}}\right)\right\|_2^2 dt \leq \int_{t\in I} \left\| \frac{dx_t^i}{dt} - h\left(x_t^i; \theta^{(r)}, \phi^{(r)}_{e_i^{(r)}}\right)\right\|_2^2 dt.$$

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

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

Now we have

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

Now consider the second part of the proposition. Define the following space of $\theta, \boldsymbol{\phi}$

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

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

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

Note that because $|\mathcal{H}| < \infty$, 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 $R_{\hat{\mathbf{e}}^{(r+1)}}(\theta^{(r+1)}, \boldsymbol{\phi}^{(r+1)}) < R_{\hat{\mathbf{e}}^{(r)}}(\theta^{(r)}, \boldsymbol{\phi}^{(r)})$, we must have $\hat{\mathbf{e}}^{(r+1)} \neq \hat{\mathbf{e}}^{(r)}$. Otherwise we will have

$$R_{\hat{\mathbf{e}}^{(r+1)}}\left(\theta^{(r+1)}, \boldsymbol{\phi}^{(r+1)}\right) = R_{\hat{\mathbf{e}}^{(r)}}\left(\theta^{(r+1)}, \boldsymbol{\phi}^{(r+1)}\right) = R_{\hat{\mathbf{e}}^{(r)}}\left(\theta^{(r)}, \boldsymbol{\phi}^{(r)}\right).$$

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

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

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

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

Therefore, we must have

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

Now the claim follows. □

B Environment Parameters

The parameters for LV, GS and NS systems are respectively given in Table 3, 4 and 5.

Table 3: 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 4: 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 5: 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	$0.1 * (\sin(2\pi(2X+Y)) + \cos(2\pi * (2X+Y)))$

C Plots on Learned Dynamics

The plots in Figure 4 depict the recovered test trajectories generated by the learned neural network. Upon close examination, it is evident that the trajectories predicted by DynaInfer closely align with those of the Oracle and the ground truth for the chosen system.

D Experimental Setup

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 [23]. Neural network architectures, optimizers, and parameters for the base models are configured as described in their respective papers.

E Environment Assignment Convergence on GS

Figure 5 presents the temporal evolution of environment assignment probabilities using the LEADS base model on GS.

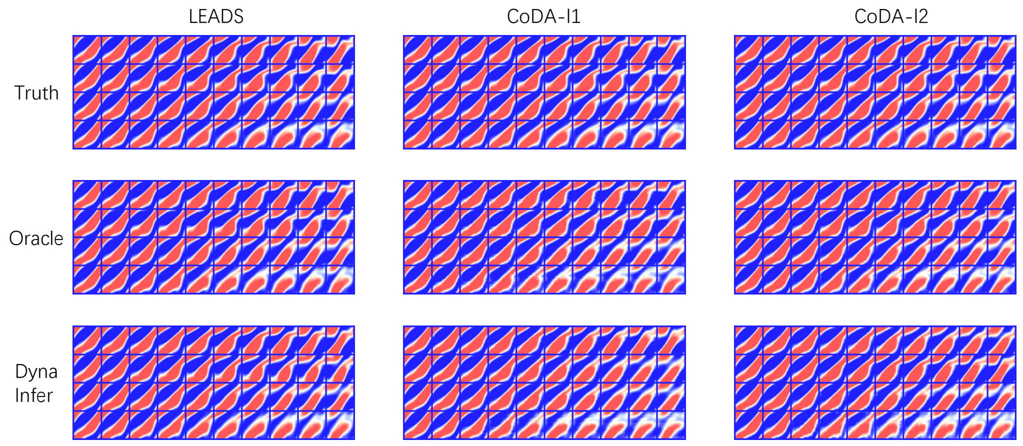

Figure 4: Comparison of final NS states predicted by DynaInfer and Oracle against the ground truth. Snapshots are arranged on a spatial 2D grid with a time interval $\Delta T = 1$.

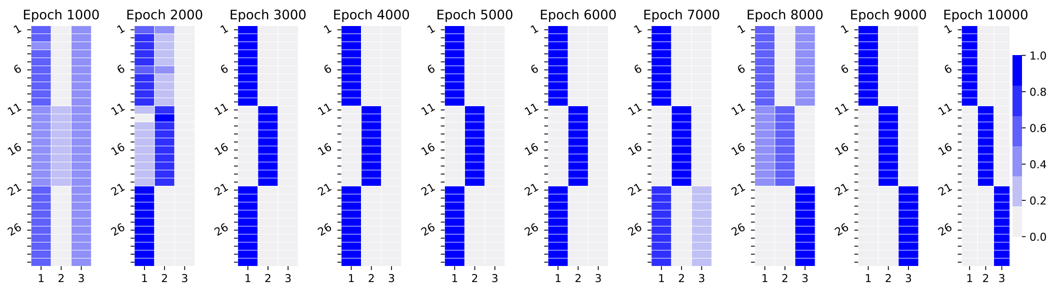

Figure 5: Environment Assignment Probability over Time with LEADS as Base Model on GS. Despite initial inaccuracies due to complex dynamics, the assignment ultimately converges to the correct label.

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 D. 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.