Neural ODE and SDE Models for Adaptation and Planning in Model-Based Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

We investigate neural ordinary and stochastic differential equations (neural ODEs and SDEs) to model stochastic dynamics in fully and partially observed environments within a model-based reinforcement learning (RL) framework. Through a sequence of simulations, we show that neural SDEs more effectively capture transition dynamics' inherent stochasticity, enabling high-performing policies with improved sample efficiency in challenging scenarios. We leverage neural ODEs and SDEs for efficient policy adaptation to changes in environment dynamics via inverse models, requiring only limited interactions with the new environment. To address partial observability, we introduce a latent SDE model that combines an ODE and a GAN-trained stochastic component in latent space. This model matches or exceeds the performance of the state-based SDE variant and outperforms ODE-based alternatives across stochastic variants of continuous control benchmarks, providing the first empirical demonstration of action-conditional latent neural SDEs for planning in such settings.

1 Introduction

In recent years, the family of neural ordinary differential equations (neural ODEs) (Chen et al., 2018; Rubanova et al., 2019; Li et al., 2020; Kidger et al., 2020; 2021) have emerged as a powerful framework for modelling dynamical systems. The general idea of these models is to use neural networks to parameterise the derivatives of the system's dynamics, while the state evolution is computed by a numerical differential equation solver. Such a way of decoupling the modelling of dynamics from the discretisation scheme leads to models with increased fidelity for capturing complex, continuous-time transitions. As a result, Neural DEs are especially well-suited for learning and representing transition dynamics in reinforcement learning (RL).

Recent work has demonstrated that integrating neural ordinary differential equations (neural ODEs) (Chen et al., 2018) and stochastic differential equations (neural SDEs) Li et al. (2020); Kidger et al. (2021) into RL frameworks can improve the modelling of continuous-time dynamics. In partially observable Markov decision processes (POMDPs) and model-free RL, recurrent neural ODEs have shown robustness to irregularly sampled observations, due to their capacity to model non-uniform time series (Zhao et al., 2023). Latent neural ODEs have also been employed as data-driven dynamics models in model-based RL, providing higher sample efficiency compared to model-free baselines (Du et al., 2020). Additionally, incorporating Neural ODEs with control-theoretic constraints has supported the development of safe and stable RL in continuous-time domains (Zhao et al., 2025).

Neural SDEs extend the neural ODE framework by jointly modelling the deterministic and stochastic components of system dynamics, allowing for uncertainty-aware policy learning. Recent studies show that offline model-based RL with neural SDEs, particularly when incorporating physics priors, can outperform state-of-the-art algorithms on low-quality datasets (Koprulu et al., 2025). Physics-constrained neural SDEs, which explicitly represent model uncertainty, have also been shown to support generalisation beyond the training distribution (Djeumou et al., 2023). While both these neural SDE frameworks could, in principle, be applied to POMDPs, practical demonstrations to date have focused on environments with full observability.

Despite recent advances, the application of latent neural SDEs to partially observed, stochastic RL environments remains underexplored. Unified frameworks that combine controlled latent neural ODEs and neural SDEs in both fully and partially observed environments are lacking, and the practical benefits of modelling stochasticity in latent dynamics for planning have not been systematically assessed. Additionally, there has been limited exploration of neural ODEs and neural SDEs in settings that require adaptation to changes in the environment.

In this work, we first demonstrate sample-efficient policy adaptation to changes in environment configuration by employing an inverse dynamics approach (Christiano et al., 2016) based on neural ODE and SDE transition models. We then introduce a unified latent ODE/SDE framework for model-based RL, motivated by the need to handle partial observability in complex, stochastic environments. The framework uses a two-phase procedure: the mean latent dynamics are learned using a latent ODE, and stochasticity is handled via a GAN-trained latent SDE. In experiments on stochastic continuous control tasks, we show that model-based RL with latent neural SDEs improves sample efficiency compared to both ODE-based models and a model-free SAC baseline in the most challenging fully or partially observed environment included in our study.

To our knowledge, there has been no prior empirical validation of action-conditioned latent neural SDEs for planning in partially observed stochastic control tasks. Our results highlight the potential benefit of latent neural SDEs for flexible, noise-aware planning in stochastic domains.

2 Background

Markov decision process (MDP) and partially observable MDP (POMDP). An MDP (Cassandra et al., 1994) is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, R \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are sets of states and actions respectively, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the probabilistic transition function (dynamics), with $\mathcal{T}(s_{t+1}|s_t, a_t)$ representing the probability of transitioning into s_{t+1} from s_t under a_t, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the deterministic reward function. The initial state s_0 follows certain distribution ρ_0, the horizon is $T \in \mathbb{N}^+ \cup \{+\infty\}$ and the discount factor is $\gamma \in [0, 1]$, with 1 for finite horizon. When the state s_t is not fully observable, MDP tuples are extended to POMDP (Cassandra et al., 1994) $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, O, R \rangle$ by adding a set of observations $\mathcal{O}$ and an emission function $O : \mathcal{S} \to \Delta(\mathcal{O})$, which probabilistically maps a state s_t to an observation o_t.

Specifically, we consider the class of POMDPs that consists of two sources of uncertainty in the transition dynamics of observed states $\mathcal{T}(o_{t+1}|o_t, a_t)$. The first source is the aleatoric stochasticity in the underlying MDP state transition $\mathcal{T}(s_{t+1}|s_t, a_t)$ that produces the observations. The second source is the epistemic uncertainty from the partial observation, which is usually reducible by taking into account a history window of partial observation and actions at each step. We assume that the emission functions O of the POMDPs are deterministic, i.e., $o_t = o(s_t)$.

In such POMDPs, we introduce a latent representation z_t that summarizes the observed history up to t $\{a_{0:t-1}, o_{0:t}\}$ as a proxy of the state s_t (Du et al., 2020; Ni et al., 2024). The transition function of the latent z_t is $\mathcal{T}(z_{t+1}|z_t, a_t, o_t)$, with the observation o_t deterministically emitted from z_t via $o_t = o(z_t)$. We are interested in learning the encoder mapping the history to the latent variable, the latent transition function and the decoder mapping the latent back to the observation, which together enable the prediction of the next observation in the POMDP.

Neural Ordinary Differential Equation (Neural ODE). Neural differential equations are a family of differential equations whose rate functions are approximated by learnable neural networks (Kidger, 2022). A classic example is a neural ODE (Chen et al., 2018):

$$\frac{\mathrm{d}s_t}{\mathrm{d}t} = f_\theta(s_t, t), \quad \text{where} \quad s_{t_0} = s_0 \tag{1}$$

where s_t denotes the state at time t as the solution of the ODE initial-value problem (IVP). The initial state at t_0 is s_0. The rate function f_θ is mostly parameterized by a multi-layer perceptron (MLP) with learnable parameters θ. The general regime of neural ODE allows any off-the-shelf numerical integrator to solve for the above ODE IVP. More concretely:

$$\hat{s}_{i+1} = \text{neuralODEsolve}(f_\theta(\hat{s}_i, t_0 + i\Delta t)), \quad i = 0, \dots, T-1 \tag{2}$$

where neural ODEsolve refers to the ODE numerical integrator used. $\hat{s}_{i+1}$ is a step forward of the neural ODEsolve from $\hat{s}_i$ with a pre-defined, fixed step size Δt.

A latent ODE (Rubanova et al., 2019) utilizes the neural ODE to generate latent trajectories in a variational autoencoder (VAE) manner (Kingma, 2013), which can be formulated as:

$$z_0 \sim q_\phi(z_0|o_{0:T}), \quad z_{i+1} = \text{neural ODEsolve}(f_\theta(z_i, t_0 + i\Delta t)), \quad \hat{o}_i = o_\theta(z_i), \quad i = 0, \ldots, T-1 \tag{3}$$

Here, an (RNN) encoder parameterized by ϕ maps observations $o_{0:T}$ to a distribution of the initial latent state z_0. A decoder parameterized by θ reconstructs observations $\hat{o}_{0:T}$ via a latent-variable ODE determined by sampled z_0 and an emission function $o_\theta(\cdot)$.

Neural Stochastic Differential Equation (Neural SDE). A neural SDE consists of a parameterized deterministic drift term and a parameterized stochastic diffusion term:

$$\mathrm{d}s_t = f_\theta(s_t, t)\mathrm{d}t + g_\theta(s_t, t)\mathrm{d}w_t, \quad \text{where} \quad s_{t_0} \sim \mu \tag{4}$$

where $\{s_t\}_{t\geq t_0}$ is a continuous-time stochastic process, whose initial state s_{t_0} is drawn from some probability distribution μ. $\{w_t\}_{t\geq 0}$ is the Brownian motion. The paper follows the idea of training a neural SDE as a (Wasserstein) generative adversarial net (GAN) (Goodfellow et al., 2014; Arjovsky et al., 2017), where the real and generated data samples are (interpolated) infinite-dimensional paths (Kidger et al., 2021).

3 Methodology

In this section, we extend the vanilla autonomous neural ODE/SDE-based models to their controlled variants by incorporating actions to model the MDP state transition dynamics $\mathcal{T}(s_{t+1}|s_t, a_t)$ and POMDP latent transition $\mathcal{T}(z_{t+1}|z_t, a_t, o_t)$. We also describe how to train a policy for planning based on the learned transition model, as well as how to adapt the model and policy to a similar environment without retraining from scratch.

3.1 Dynamics model learning

We use the terms neural ODE/SDE and latent ODE to overload the names of original autonomous models and here denote their controlled variants. In a similar vein to latent neural ODEs, we also propose the latent SDE by employing the neural SDE in the encoded latent space, which, to the best of our knowledge, has not been proposed in the literature.

Neural ODE. Let the MDP transition dynamics of a deterministic environment be defined as an autonomous neural ODE (t is not explicitly given as an argument to f_θ):

$$\hat{s}_{i+1} = \text{neuralODEsolve}(f_\theta(\hat{s}_i, a_i)), \quad i = 0, \ldots, T-1 \tag{5}$$

We optimize the parameters θ by minimizing the following mean squared error (MSE) between the predicted $\hat{s}_{0:T}$ and observed state trajectories $s_{0:T}$:

$$\min_\theta \mathbb{E}_{(s_{0:T}, a_{0:T-1})\sim\mathcal{D}_m} \left[\sum_{i=0}^{T} \|\hat{s}_i - s_i\|_2^2\right] \tag{6}$$

where real state-action trajectories are sampled from a replay buffer $\mathcal{D}_m$.

Latent ODE. Similar to Du et al. (2020), we modify the vanilla latent ODE in Eq. 3 to model the POMDP latent transition by taking actions a_i and (predicted) observations $\tilde{o}_i$ into account for the underlying evolution of RNN hidden states h_i in the encoder and ODE latent states z_i in the decoder. Specifically, the adjusted latent ODE consists of the following RNN encoder parameterized by ϕ:

$$\begin{gathered} h_{i+1} = \text{RNNCell}_\phi(h_i, a_i, \tilde{o}_i), \quad \hat{o}_i = o_\phi(h_i), \quad i = 0, \ldots, T-1, \\ [\mu_{z_0}, \sigma_{z_0}] = \text{MLP}_\phi(h_T), \quad q_\phi(z_0|o_{0:T}, a_{0:T-1}) = \mathcal{N}(z_0; \mu_{z_0}, \text{diag}(\sigma_{z_0}^2)) \end{gathered} \tag{7}$$

and neural ODE decoder parameterized by θ:

$$\begin{gathered} z_0 \sim q_\phi(z_0|o_{0:T}, a_{0:T-1}), \quad \tilde{z}_i = \text{MLP}_\theta(z_i, a_i, \tilde{o}_i), \quad z_{i+1} = \text{neural ODEsolve}(f_\theta(\tilde{z}_i)), \\ \hat{o}_i = o_\theta(z_i), \quad p_\theta(o_{i+1}|z_i, a_i, \tilde{o}_i) = \mathcal{N}(o_{i+1}; o_\theta(z_{i+1}), I), \quad i = 0, \ldots, T-1 \end{gathered} \tag{8}$$

where $o_\theta(\cdot)$ denotes the parameterized emission function, $\tilde{z}_i$ has the same dimension as z_i and represents the transformed latent state at which the derivative of latent ODE f_θ is evaluated. $\tilde{o}_i$ could be either the real observation o_i or predicted observation $\hat{o}_i$ during training, while always setting $\tilde{o}_i = \hat{o}_i$ during inference. In this paper, models are typically trained using teacher forcing, i.e., $\tilde{o}_i = o_i$, as this has been shown to be more sample-efficient and effective for training.

The above latent ODE is trained end-to-end by maximizing the evidence lower bound (ELBO) over observation trajectories $o_{0:T}$:

$$\max_{\phi,\theta} \mathbb{E}_{(o_{0:T}, a_{0:T-1})\sim\mathcal{D}_m}\Bigg[\mathbb{E}_{z_0\sim q_\phi(z_0|o_{0:T},a_{0:T-1})}\Bigg[\sum_{i=0}^{T-1}\log p_\theta(o_{i+1}|z_i,a_i,\tilde{o}_i)\Bigg] - D_{\mathrm{KL}}\left(q_\phi(z_0|o_{0:T},a_{0:T-1}) \parallel p(z_0)\right)\Bigg] \tag{9}$$

Here, the joint generative distribution is decomposed as $p_\theta(o_{0:T}|z_{0:T}, a_{0:T-1}) = \prod_{i=0}^{T} p_\theta(o_{i+1}|z_i, a_i, \tilde{o}_i)$ and $p(z_0) = \mathcal{N}(z_0; 0, I)$ is the standard Gaussian prior.

Neural SDE. We employ a GAN-based SDE to generate synthetic trajectories that closely match those sampled from a stochastic MDP transition. More precisely, the adapted neural SDE trained as Wasserstein GAN (WGAN) consists of a neural SDE generator parameterized by θ:

$$\hat{s}_{i+1} = \text{neural SDEsolve}(f_\theta(\tilde{s}_i, a_i), g_\theta(\tilde{s}_i, a_i), \Delta w_i), \quad \bar{s}_i = \mathrm{MLP}_\theta(\hat{s}_i) \quad i = 0, \ldots, T-1 \tag{10}$$

and an MLP critic (discriminator) parameterized by ψ:

$$y = \mathrm{MLP}_\psi(\bar{s}_{0:T}, a_{0:T-1}) \tag{11}$$

Here, the projection from $\hat{s}_i$ to $\bar{s}_i$ allows for more flexible generated states. Different from Kidger et al. (2021) that uses a neural controlled differential equation (CDE) as a discriminator, we simply use an MLP that takes flattened state-action trajectories as input. This change makes optimization much easier, but also limits the discriminator's capacity for processing varying-length trajectories.

Let $G_\theta : (w_{0:T}, a_{0:T-1}) \to s_{0:T}$ denote the overall map of the generator from paths of noises and actions to that of synthetic observations, and $D_\psi : (s_{0:T}, a_{0:T-1}) \to y$ denote the overall map of the critic from paths of real/synthetic observations and actions to a scalar score. The training dynamics of the WGAN are formulated as follows:

$$\min_{\theta} \max_{\substack{\psi \\ \|D_\psi\|_L \le 1}} \mathbb{E}_{(s_{0:T},a_{0:T-1})\sim\mathcal{D}_m}[D_\psi(s_{0:T}, a_{0:T-1})] - \mathbb{E}_{\substack{w_{0:T}\sim p_w \\ a_{0:T-1}\sim\mathcal{D}}}[D_\psi(G_\theta(w_{0:T}, a_{0:T-1}), a_{0:T-1})] \tag{12}$$

where $\|D_\psi\|_L \le 1$ represent the Lipschitz continuity constraint enforced on the critic function D_ψ. We use the gradient penalty (Gulrajani et al., 2017) to achieve the Lipschitz constraint.

Latent SDE. A weakness of latent ODE is that it can only model deterministic MDP transitions (or the mean dynamics of a stochastic MDP transitions) of states underlying a POMDP. To learn the full stochastic MDP state transition, we employed a phased training framework. The latent ODE (Eqs. 7 and 8) is first employed to capture the mean transition dynamics in the latent space. Specifically, the encoder-decoder pair of the latent ODE is described as follows:

$$z_0 \sim q_\phi(z_0|o_{0:T}, a_{0:T-1}), \quad z_{i+1} = \text{neural ODEsolve}(f_{\theta_\mu}(z_i, a_i, \tilde{o}_i)), \quad \hat{o}_i = o_{\theta_\mu}(z_i), \quad i = 0, \ldots, T-1 \tag{13}$$

where the recurrent encoder and the ODE decoder are respectively parameterized by ϕ and θ_μ, which are trained in an end-to-end VAE-style setup (Eq. 9).

We then freeze the learned ODE decoder and deploy it as the drift function of an SDE in the latent space. The diffusion function of the SDE and the emission function are further employed to model the variance of

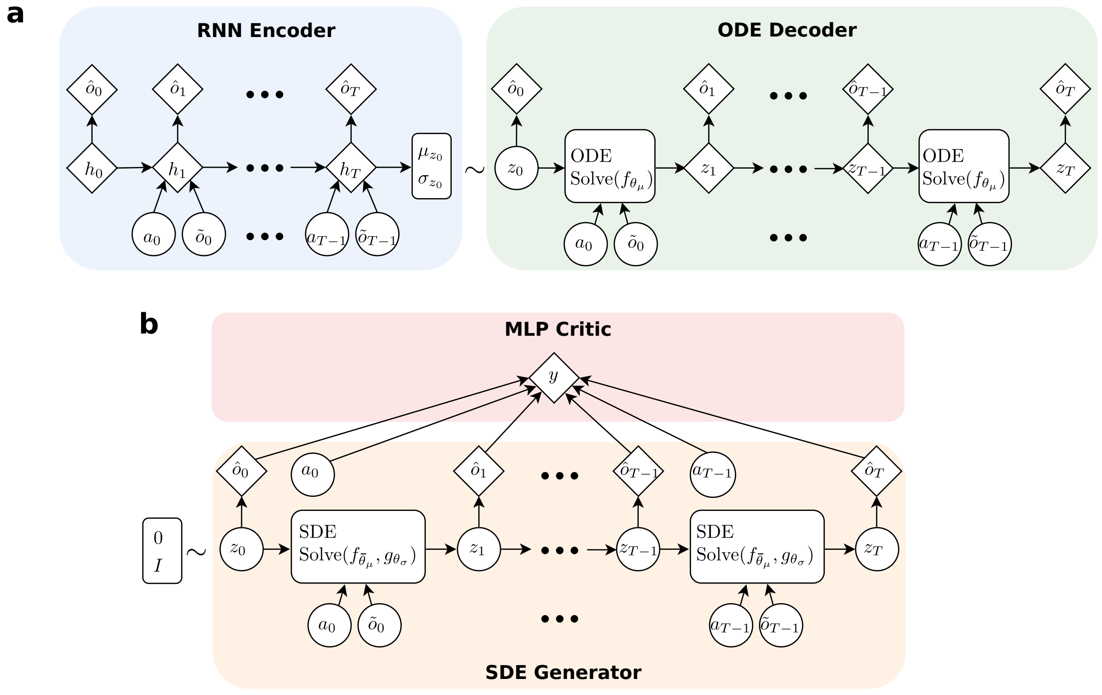
a
RNN Encoder
ODE Decoder
ODE Solve(f_{θ_μ})
b
MLP Critic
SDE Solve($f_{\bar{\theta}_\mu}, g_{\theta_\sigma}$)
SDE Generator

Figure 1: Computational graph of the latent SDE. **(a)** The RNN encoder and ODE decoder from the latent ODE. The encoder is only utilized during the model training. In the inference (generation) stage, the initial latent z_0 is sampled from the standard Gaussian prior distribution instead. Note that the $\tilde{o}_i$ could denote either the real observation o_i or predicted one $\hat{o}_i$ during training, depending on whether teacher forcing strategy is used, yet it is always set to be $\tilde{o}_i = \hat{o}_i$ during inference. **(b)** The SDE generator and MLP critic of the latent SDE. In the SDE solver, we use the learnt drift function $f_{\bar{\theta}_\mu}$ from the ODE decoder in **(a)** and train the diffusion function g_{θ_σ} to capture the full stochasticity in the latent space. In both panels, the deterministic variables are represented as diamonds while the stochastic variables are depicted as circles.

the latent transition dynamics and the latent-to-observation projection, respectively. Putting together, we have the following equations:

$$z_0 \sim p(z_0), \quad z_{i+1} = \text{neural SDEsolve}(f_{\bar{\theta}_\mu}(z_i, a_i, \tilde{o}_i), g_{\theta_\sigma}(z_i, a_i, \tilde{o}_i), \Delta w_i) \quad \hat{o}_i = o_{\theta_\sigma}(z_i), \quad i = 0, \ldots, T-1,$$
$$y = \text{MLP}_\psi(\hat{o}_{0:T}, a_{0:T-1}) \tag{14}$$

where z_0 is sampled from the Gaussian prior $p(z_0)$ used in the VAE training. $f_{\bar{\theta}_\mu}$ and g_{θ_σ} are the drift and diffusion function of the SDE, where $\bar{\theta}_\mu$ denotes the detached optimal θ_μ from the first phase. The diffusion g_{θ_σ} and emission function o_{θ_σ} of the SDE generator are trained in pairs with the MLP critic parameterized by ψ in GAN-style setup (Eq. 12).

We term this model framework that combines latent ODE and neural SDE as latent SDE, and depict its computation graph in Fig. 1.

3.2 Planning and policy learning with learned dynamics

When addressing sample-intensive tasks (such as Mujoco), a model-free RL method is often trained off-policy using near-optimal transitions collected by a planning strategy driven by learned, task-specific dynamics, thereby reducing sample complexity. In this paper, following Du et al. (2020), we adopt a framework that combines model predictive control (MPC) (Nagabandi et al., 2017; Chua et al., 2018), which searches for the most promising exploratory action, with soft actor-critic (SAC) (Haarnoja et al., 2018), which learns the optimal policy (actor) and value function (critic).

Algorithm 1: MPC with SAC for POMDPs.

```
Input  : The empty replay buffer for model learning $\mathcal{D}_m$ and for policy learning $\mathcal{D}_p$, the initial latent
         transition model $\mathcal{T}_\theta(z_{t+1}|z_t, a_t, o_t)$, the reward function $R(o_t, a_t, o_{t+1})$, the initial actor
         $\pi_\omega(a_t|o_t)$, the two critics $Q^\pi_{\varphi_i}(o_t, a_t)$ and their target networks $Q^\pi_{\varphi'_i}(o_t, a_t),\ i = 1, 2$.
Params: The planning horizon $H$, the search population $K$, the number of environment steps $M$ and
         the number of epochs $E$.
Output: Learned $P_\theta$, $\pi_\omega$ and $\min_{i=1,2} Q^\pi_{\varphi_i}$.
Collect trajectories using a uniformly distributed policy and save them into $\mathcal{D}_m$;
for $i = 1$ to $E$ do
    Update $\mathcal{T}_\theta(z_{t+1}|z_t, a_t, o_t)$ using data from $\mathcal{D}_m$, as detailed in Section 3.1;
    Observe the initial state $o_0$ and initialize the latent state $z_0$ ;
    for $j = 1$ to $M$ do
        for $k = 1$ to $K$ do
            $\hat{o}^{(k)}_0, z^{(k)}_0 \leftarrow o_{j-1}, z_{j-1}$;
            for $h = 1$ to $H$ do
                Select the action $a^{(k)}_{h-1} \sim \pi_\omega(\cdot|\hat{o}^{(k)}_{h-1})$;
                Transit to next latent $z^{(k)}_h \leftarrow \mathcal{T}_\theta(\cdot|z^{(k)}_{h-1}, a^{(k)}_{h-1}, \hat{o}^{(k)}_{h-1})$ and emit the observation
                 $\hat{o}^{(k)}_h \leftarrow o_\theta(z^{(k)}_h)$;
                Calculate the reward $\hat{r}^{(k)}_{h-1} \leftarrow R(\hat{o}^{(k)}_{h-1}, a^{(k)}_{h-1}, \hat{o}^{(k)}_h)$;
            end
            Select the action $a^{(k)}_H \sim \pi_\omega(\cdot|\hat{o}^{(k)}_H)$;
        end
        Select the index of the sequence with the maximum return among the $K$ sequences:
         $k^* \leftarrow \arg\max_k \sum_{h=1}^{H} \gamma^{h-1} \hat{r}^{(k)}_{h-1} + \gamma^H \min_{i=1,2} Q^\pi_{\varphi_i}(\hat{o}^{(k)}_H, a^{(k)}_H)$;
        Select the first action of the best sequence $a_{j-1} \leftarrow a^{(k^*)}_0$;
        Execute $a_{j-1}$ and observe the next state $o_j$;
        Transit to the next latent $z_j \leftarrow \mathcal{T}_\theta(\cdot|z_{j-1}, a_{j-1}, o_{j-1})$;
        Calculate the reward $r_{j-1} \leftarrow R(o_{j-1}, a_{j-1}, o_j)$;
        if $o_j$ is not the terminal state then
            Store $(o_{j-1}, a_{j-1}, r_{j-1}, o_j)$ in $\mathcal{D}_p$;
        else
            Store $(o_{j-1}, a_{j-1}, r_{j-1}, \mathrm{NULL})$ in $\mathcal{D}_p$, reset the environment;
            Observe the initial state $o_0$ and initialize the latent state $z_0$ ;
            continue;
        end
        Update $Q^\pi_{\varphi_1}$, $Q^\pi_{\varphi_2}$ and $\pi_\omega$ using data from $\mathcal{D}_p$;
        Update target networks $Q^\pi_{\varphi'_1}$, $Q^\pi_{\varphi'_2}$;
    end
    Store the trajectories collected using the current actor in $\mathcal{D}_m$;
end
```

At each time step t, MPC simulates K trajectories over a planning horizon H, where actions and next states are sampled from the actor and the transition model, respectively. To prevent the model rollouts from becoming shortsighted, the critic estimates the cumulative reward beyond the planning horizon. This estimate, together with the cumulative reward from the MPC rollouts, forms the return for the state at time t. Subsequently, the first action of the trajectory that achieves the highest return is selected to interact with the environment.

This approach interleaves training of the transition model (when needed), data collection, and optimization of both the policy and the value function, thereby enabling policy learning from a limited number of data samples. The complete algorithm for POMDPs, adapted from Du et al. (2020), is provided in Algorithm 1, with the MPC planning component highlighted in blue.

3.3 Model and Policy Adaptation

In this subsection, we introduce an efficient way to adapt the learned transition model and policy of a source environment to a target environment, at the cost of a minimal amount of data from the target environment. The source environment typically refers to a simulated or controlled setting while the target environment denotes the real-world or modified simulated scenario, usually presenting novel dynamics or disturbances. Inspired by prior work by Christiano et al. (2016), we employ an adaptation architecture based on an inverse dynamics model, which allows us to leverage the high-level characteristics of the source policy while adapting to the specifics of the target domain.

In what follows, we use the superscripts "src" and "tge" to denote variables and models associated with the source and target environments, respectively. We assume that both environments are characterized by MDP transitions and share the same actuated degrees of freedom. As illustrated in Fig. 2A, given a current target state s_t^{tge}, we first compute the corresponding source action $a_t^{\mathrm{src}} = \pi^{\mathrm{src}}(s_t^{\mathrm{tge}})$ using the predefined source policy π^{src}. Next, we estimate the subsequent source state $\hat{s}_{t+1}^{\mathrm{src}} = \mathcal{T}_\theta^{\mathrm{src}}(s_t^{\mathrm{tge}}, a_t^{\mathrm{src}})$ via the parameterized source transition model $\mathcal{T}_\theta^{\mathrm{src}}$, conditioned on the current state s_t^{tge} and action a_t^{src}. Finally, the current target action $a_t^{\mathrm{tge}} = I_\eta(s_t^{\mathrm{tge}}, \hat{s}_{t+1}^{\mathrm{src}})$ is computed using the inverse dynamics model I_η, which maps the current target state s_t^{tge} and the desired next state $\hat{s}_{t+1}^{\mathrm{src}}$to a target action that drives the next target state towards the desired state $\hat{s}_{t+1}^{\mathrm{src}}$ as close as possible.

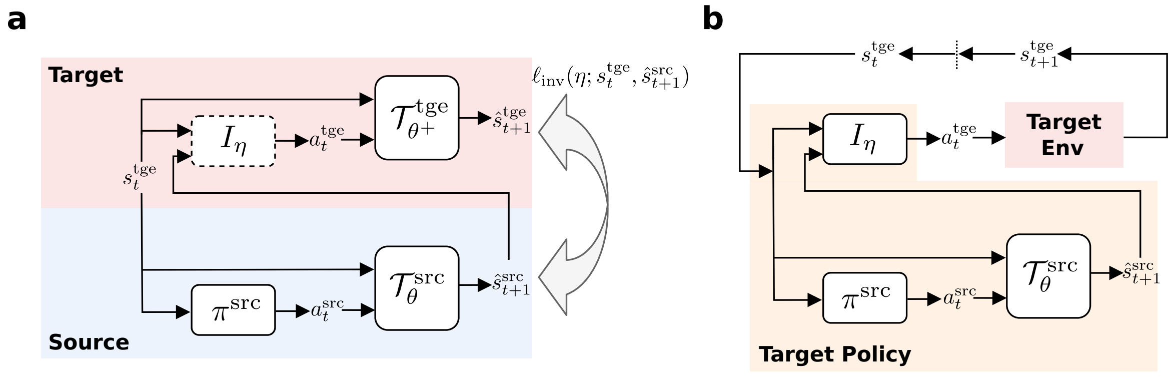
a
Target
Source
I_η
a_t^{tge}
$\mathcal{T}_{\theta^+}^{\mathrm{tge}}$
$\hat{s}_{t+1}^{\mathrm{tge}}$
s_t^{tge}
π^{src}
a_t^{src}
$\mathcal{T}_\theta^{\mathrm{src}}$
$\hat{s}_{t+1}^{\mathrm{src}}$
$\ell_{\mathrm{inv}}(\eta; s_t^{\mathrm{tge}}, \hat{s}_{t+1}^{\mathrm{src}})$
b
s_t^{tge}
s_{t+1}^{tge}
I_η
a_t^{tge}
Target Env
π^{src}
a_t^{src}
$\mathcal{T}_\theta^{\mathrm{src}}$
$\hat{s}_{t+1}^{\mathrm{src}}$
Target Policy

Figure 2: Overview of the architecture for training and deploying a policy adapted from the source domain in the target domain. **(a)** Given an off-the-shelf source policy π^{src} and transition model $\mathcal{T}_\theta^{\mathrm{src}}$, we use an inverse dynamics model I_η to generate the target action a_t^{tge} leading to the next target state $\hat{s}_{t+1}^{\mathrm{tge}}$. I_η is trained via minimizing the mismatch between the predicted next state $\hat{s}_{t+1}^{\mathrm{tge}}$ and the desired state $\hat{s}_{t+1}^{\mathrm{src}}$. **(b)** The trained inverse model, combined with the source policy and transition, is deployed in the real target environment as an adapted target policy.

Data collection/training. In order to minimize the mismatch between the next target state led by the target action and the anticipated next state, we need a differentiable transition model for the real transition dynamics in the target domain. Specifically, we adapt the deterministic component of the source transition model parameterized by θ to that of the target transition model parameterized by the augmented parameter set $\theta^+ = \theta \cup \theta_{\mathrm{aug}}$, which includes additional parameters θ_{aug} that capture the variation in the deterministic component of transition dynamics between the source and target environments. We only train the additional parameters θ_{aug} of the augmented neural net modeling the deterministic component of the target transition, using a small set of data using random policy from the target environment saved in the buffer $\mathcal{D}_m$. Similar to the loss for a deterministic Neural ODE defined in Eq. 6, we minimize the following MSE loss between the predicted and real target state trajectories:

$$\mathcal{L}_{\mathrm{m}}(\theta_{\mathrm{aug}}; s_{0:T}^{\mathrm{tge}}, a_{0:T-1}^{\mathrm{tge}}) = \mathbb{E}_{(s_{0:T}^{\mathrm{tge}}, a_{0:T-1}^{\mathrm{tge}}) \sim \mathcal{D}_m} [\ell_{\mathrm{m}}(\theta_{\mathrm{aug}}; s_{0:T}^{\mathrm{tge}}, a_{0:T-1}^{\mathrm{tge}})] \tag{15}$$

where

$$\ell_{\mathrm{m}}(\theta_{\mathrm{aug}}; s^{\mathrm{tge}}_{0:T}, a^{\mathrm{tge}}_{0:T-1}) = \sum_{t=1}^{T} \|\hat{s}^{\mathrm{tge}}_t - s^{\mathrm{tge}}_t\|_2^2 = \sum_{t=1}^{T} \|\mathcal{T}^{\mathrm{tge}}_{\theta^+}(\hat{s}^{\mathrm{tge}}_{t-1}, a^{\mathrm{tge}}_{t-1}) - s^{\mathrm{tge}}_t\|_2^2 \tag{16}$$

Empirically we only require a small amount of data from the target domain to train the target transition model augmented from the source transition model, compared with training a target transition model from scratch. We freeze the target transition model once it is sufficiently accurate, and then use it for training the inverse dynamics model, which is interleaved with data collection. Specifically, we repeat the following data collection/training loop: during the data collection phase, given a current target state, we use the preliminary source policy and transition model to obtain the desired next state, and then use the learned-so-far inverse dynamics model to produce the target action, which interacts with the real target environment leading to the next target state. We repeat the target environment steps and save the sequence along the steps $\{\cdots, s^{\mathrm{tge}}_t, \hat{s}^{\mathrm{src}}_{t+1}, s^{\mathrm{tge}}_{t+1}, \hat{s}^{\mathrm{src}}_{t+2}, \cdots\}$ into the buffer $\mathcal{D}_{inv}$. Target states along the trajectory collected in such an on-the-fly fashion are near the optimal target trajectory, which will lead to much faster convergence of the inverse dynamics model compared with training data collected by a random policy. During the training phase, we optimize the following mean squared error (MSE) loss between the predicted next target state and the desired next state, using a mini-batch of real-desired transition pairs sampled from $\mathcal{D}_{inv}$:

$$\mathcal{L}_{\mathrm{inv}}(\eta; s^{\mathrm{tge}}_t, \hat{s}^{\mathrm{src}}_{t+1}) = \mathbb{E}_{(s^{\mathrm{tge}}_t, \hat{s}^{\mathrm{src}}_{t+1}) \sim \mathcal{D}_{inv}}[\ell_{\mathrm{inv}}(\eta; s^{\mathrm{tge}}_t, \hat{s}^{\mathrm{src}}_{t+1})] \tag{17}$$

where

$$\ell_{\mathrm{inv}}(\eta; s^{\mathrm{tge}}_t, \hat{s}^{\mathrm{src}}_{t+1}) = \|\hat{s}^{\mathrm{tge}}_{t+1} - \hat{s}^{\mathrm{src}}_{t+1}\|_2^2 = \|\mathcal{T}^{\mathrm{tge}}_{\theta^+}(s^{\mathrm{tge}}_t, I_\eta(s^{\mathrm{tge}}_t, \hat{s}^{\mathrm{src}}_{t+1})) - \hat{s}^{\mathrm{src}}_{t+1}\|_2^2 \tag{18}$$

It is worth noting that the above MSE loss not only applies to the next states generated by deterministic transition dynamics but also to the means of the next states when the transition dynamics are stochastic. The averaged dynamics of the stochastic transition described by an SDE (i.e., the drift term of the SDE) can be captured by the Neural ODE model. Therefore we are only interested to adapt the model and policy when the mean-field (drift) dynamics vary between the source and target domain as the diffusion dynamics capture the aleatoric uncertainty, which is irreducible (Chua et al., 2018; Han et al., 2024).

Deployment. As illustrated in Fig. 2B, we can use the learned inverse dynamics model as a target policy, which is computed via adaptation from the source policy as follows:

$$\pi^{\mathrm{tge}}(s^{\mathrm{tge}}_t) = I_\eta(s^{\mathrm{tge}}_t, \mathcal{T}^{\mathrm{src}}_\theta(s^{\mathrm{tge}}_t, \pi^{\mathrm{src}}(s^{\mathrm{tge}}_t))) \tag{19}$$

4 Experiments

We empirically evaluate our ODE and SDE-based models across three continuous and stochastic environments with increasing complexity. In section 4.1, we show that the neural SDE characterizes a more accurate stochastic transition than the neural ODE, which only learns the deterministic drift function of the neural SDE that corresponds to the averaged dynamics of the stochastic transition. In section 4.2, we then demonstrate that the neural ODE/SDE allows us to adapt the policy to changes in the environment via the inverse model with fewer data. In section 4.3, we also demonstrate that SDE-based planning achieves better performances than ODE-based, in stochastic environments with both full and partial observations. Moreover, SDE-based planning is more sample-efficient than the model-free policy in the most challenging stochastic environment we tested.

Environments. For evaluation of our methods, we modify the standard deterministic OpenAI Gym environments (Towers et al., 2024): Cartpole from the classic control task, swimmer and hopper from the Mujoco locomotion task into stochastic versions by adding noises to their MDP transition dynamics. In each modified environment, both the action and observation spaces are continuous. When fully observable, the observation space includes the positions (or angles) and velocities (or angular velocities) of every degree of freedom.

- **Stochastic cartpole.** The goal is to balance a pole on a moving cart by applying forces on the cart at each step. We convert the original discrete actions to continuous ones and apply independent and identically distributed (i.i.d.) standard Gaussian force to the cart at each step, which can be formulated as an SDE of cart velocity.
- **Stochastic swimmer.** A 3-link swimmer is propelled forward in a fluid, with a 10-dimensional observation space and 2 actuators. We introduce i.i.d. Gaussian noise sampled from $\mathcal{N}(0, 10^4)$ to the stiffness parameter of the third joint of the Swimmer.
- **Stochastic hopper.** A single-legged robot is made to hop forward as far as possible, with a 12-dimensional observation space and 3 actuators. We apply stochastic winds to the hopper at each step. The magnitude of the wind is i.i.d. sampled from $\mathcal{N}(0, 10^6)$.

We further hide the position and velocity features from the observation space of the Mujoco to evaluate learning in POMDPs. It is worth noting that the POMDPs considered here satisfy our assumptions on the aleatoric and epistemic nature of the uncertainties in the POMDP transition dynamics, as described in 2. Specifically, in the partially observable stochastic swimmer, we mask the positions and angle of the front tip (**stochastic swimmer (no position)**), and its positional and angular velocities (**stochastic swimmer (no velocity)**) from the observation space (corresponding to the first three and the 5th to 8th dimensions of the observation respectively). Similarly in the partially observable stochastic hopper, we mask the positions the torso (**stochastic hopper (no position)**), and its angular velocity (**stochastic hopper (no velocity)**), which correspond to the first two and the 8th dimensions of the observation, respectively.

Baselines. We compare the performance of our proposed ODE-based and SDE-based models in the stochastic environments, in terms of model learning and model-based policy learning. In the simpler cartpole environment, we test the capacity of the neural ODE (**N-ODE**) and neural SDE (**N-SDE**) as a proxy of the real stochastic transition for model-based policy learning and adaptation. In the Mujoco environment, apart from the neural ODE/SDE, we also evaluate the latent ODE (**L-ODE**) and latent SDE (**L-SDE**) for planning and policy learning. In all stochastic environments, we use Soft Actor Critic (SAC) Haarnoja et al. (2018) as a model-free policy baseline for comparison. The SAC trains a Gaussian policy but instead uses a deterministic policy (the mean of the Gaussian) during evaluation.

4.1 Neural ODE/SDE modelling transition dynamics

Here we demonstrate the capacity of the neural ODE and SDE in mimicking the stochastic MDP transitions of the stochastic cartpole environment. We use the learned transition models as a proxy for the real transition dynamics. A model-free agent can therefore be trained in the approximated transition dynamics without interacting with the real environment. To achieve the oracle performance given by the agent trained in the real environment, the transition model should be accurate enough to recover the distribution of the possible next states.

Neural ODEs model the mean of stochastic transitions while Neural SDEs capture the full stochastic dynamics. Fig. 3a depicts the evolution of marginal distributions of cart velocity across time predicted by the neural ODE and SDE, in comparison with the real marginal distribution. The ODE-based distributions are peakier at the mean than the real distribution and fail to recover values away from the mean. On the other hand, the SDE-based distributions mostly cover the real distributions. In addition, Fig. 3b shows the sample paths from the ODE and SDE-based distributions against the paths from the real distributions. Once again the ODE-based paths capture only the averaged tendency of the real paths, while SDE-based paths show better agreement with the real ones. For other features in the observation space of the stochastic cartpole, since their transition dynamics are deterministic, the neural SDE learns almost the same transition dynamics as the neural ODE (see Fig. 6 and 7 for distribution and path matching respectively in the Appendix).

We evaluate the performance of agents trained on modeled transition dynamics in the real environment (Fig. 3c). Performance of SDE-based policy is much closer to the near-optimal performance of the model-free oracle policy than the ODE-based policy, in terms of the resemblance of their distribution of returns to the oracle distribution. The poor performance of ODE-based policy when deployed in the actual environment is due to the low fidelity of the model used for training the policy.

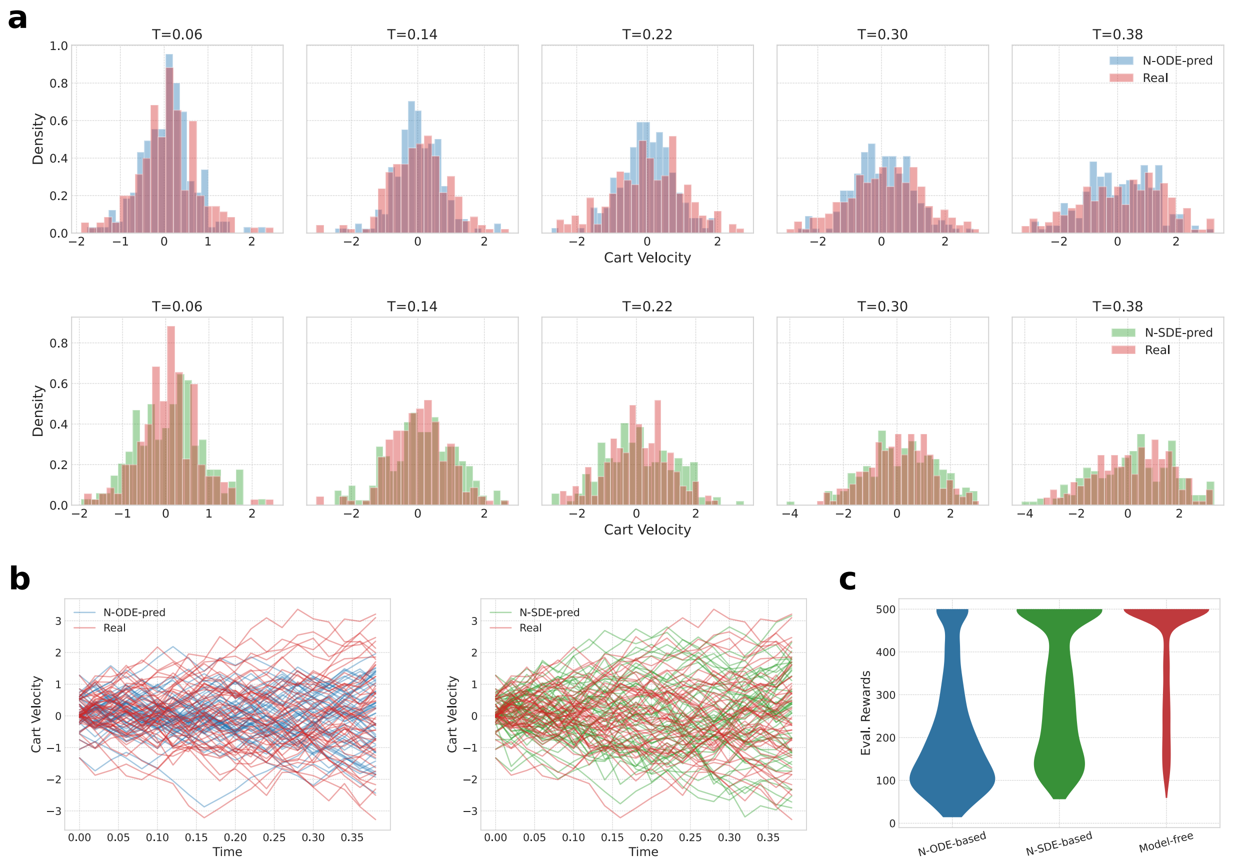

Figure 3: Comparison between neural ODE and SDE on stochastic cartpole. **(a)** Predicted histograms of cart velocity at 16%, 37%, 58%, 79%, 100% of the total timesteps by the neural ODE and SDE against the real histograms. **(b)** 50 sample paths from the distributions predicted by neural ODE and SDE against 50 paths from the real distributions. The neural SDE learns to match the real distributions and sample paths better than the neural ODE. **(c)** Policies trained in the approximate transition dynamics of neural ODE (N-ODE-based) and SDE (N-SDE-based), as well as in real transition (model-free), are evaluated in the real environment. N-SDE-based policy achieves similar performance to the model-free policy (used as the oracle here).

4.2 Policy adaptation via inverse dynamics model

In this section, we empirically test the policy adaptation method described in section 3.3 in both deterministic and stochastic Cartpole environments with increasing pole length. The results show the improvement of our adapted source policies based on Neural ODEs/neural SDEs over the non-adapted source policies in the target environment, and their greater sample efficiency compared with new target policies trained from scratch, given a limited interaction with the target environment.

Deterministic transition. Here, we consider the scenario where the transition dynamics of the Cartpole is deterministic. We evaluate the performance of adapted and non-adapted policies (blue and red curves in Fig. 4a) against the increment in pole length. Particularly, the adapted policy is based on Neural ODEs, which are used to model the transition dynamics in both source and target domains given their determinism. To show the sample efficiency of our ODE-adapted policy, we also compare it with a new policy trained from scratch (green curve in Fig. 4a), using the same number of target environment samples as used for training the augmented target neural ODE. The results show that the ODE-adapted policy consistently outperforms the non-adapted baseline, yet is always significantly better than the trained-from-scratch baseline when the variation between the source and target environment is not too large. Besides, the performance of both

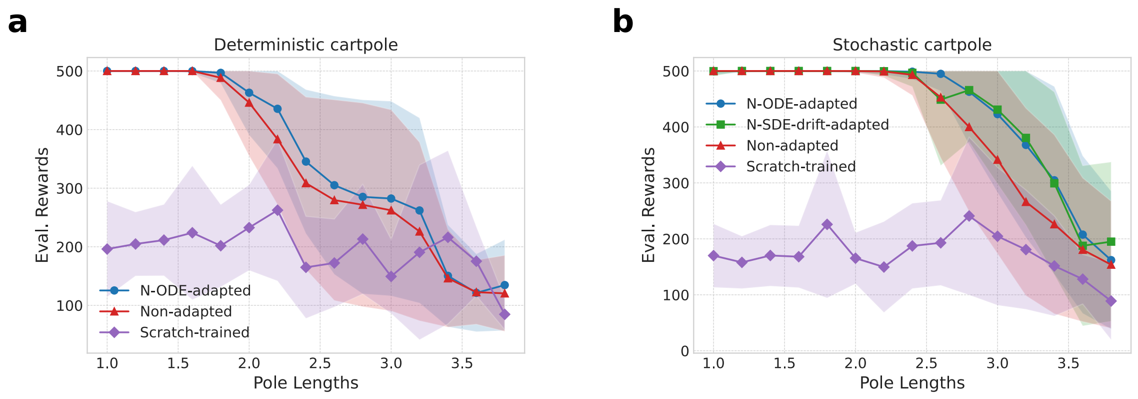

Figure 4: Evaluated performances of different policies for target Cartpole environments with increasing pole lengths (pole length is 1.0 in the source environment). The shaded regions represent the standard deviation of the evaluated returns over 5 runs. **(a)** When both the source and target transition dynamics are deterministic, we observe that within the pole length range of 1.8 to 3.2, the ODE-adapted policy consistently outperforms the original, non-adapted one, and both largely outperform the trained-from-scratch policy. **(b)** When both the source and target transition dynamics are stochastic by introducing zero-mean Gaussian noises to the velocity of the cart, we observe that within the pole length range of 2.6 to 3.8, both the ODE-adapted and SDE-drift-adapted policies achieve the similar best performances, followed by the non-adapted policy and finally the trained-from-scratch policy.

ODE-adapted and non-adapted policies declines as the pole length increases, likely due to the limited action force being insufficient to counteract the increased gravitational torque from the longer, unbalanced pole.

Stochastic transition. In the stochastic Cartpole, both the source and target transition share the same level of stochasticity, with pole length being the only varying factor to be adapted. The ODE-adapted and SDE-drift-adapted (blue and purple curves in Fig. 4b) policies respectively use the augmented Neural ODEs and the augmented drift function of Neural SDEs to model the averaged dynamics of the stochastic transition in the target environment. Both adapted policies achieve close performance, which suggests that, in a stochastic environment, the Neural ODE effectively captures the same deterministic component of the transition dynamics as the drift of the Neural SDE. Similar to the deterministic scenario, the adapted policies always outperform the non-adapted and trained-from-scratch baselines, despite the decline in their performance as the pole length increases.

Interestingly, the source policy trained in a stochastic environment (red curve in Fig. 4b) is more robust to the change to the environment than the source policy trained in a deterministic environment (red curve in Fig. 4b). This is likely because the stochasticity in the environment drives the policy to explore the state-action space thoroughly to counteract the uncertainty, which finally leads to a more robust policy. Furthermore, target policies adapted from a better source policy have a larger performance boost versus the non-adapted source policy baseline (compare the differences between the adapted and non-adapted policies in Fig. 4a with those in Fig. 4b).

4.3 Planning and policy learning with ODE/SDE-based dynamics

For more complicated stochastic Mujoco environments, we employ a framework that interleaves the exploration by MPC planning and model-free policy learning, as detailed in section 3.2. We compare the learning performance of the policies derived from MPC planning that respectively use neural ODE/SDE and their latent variants as the transition model with full and partial observations.

SDE-based policies outperform ODE-based policies across all stochastic Mujoco environments. Fig. 5 illustrates the learning curves for all baselines in the stochastic swimmer and hopper environments. SDE-based policies consistently converge more rapidly and achieve superior solutions compared to ODE-based policies. The SDE-based models explicitly represent both the mean and variance of the stochastic

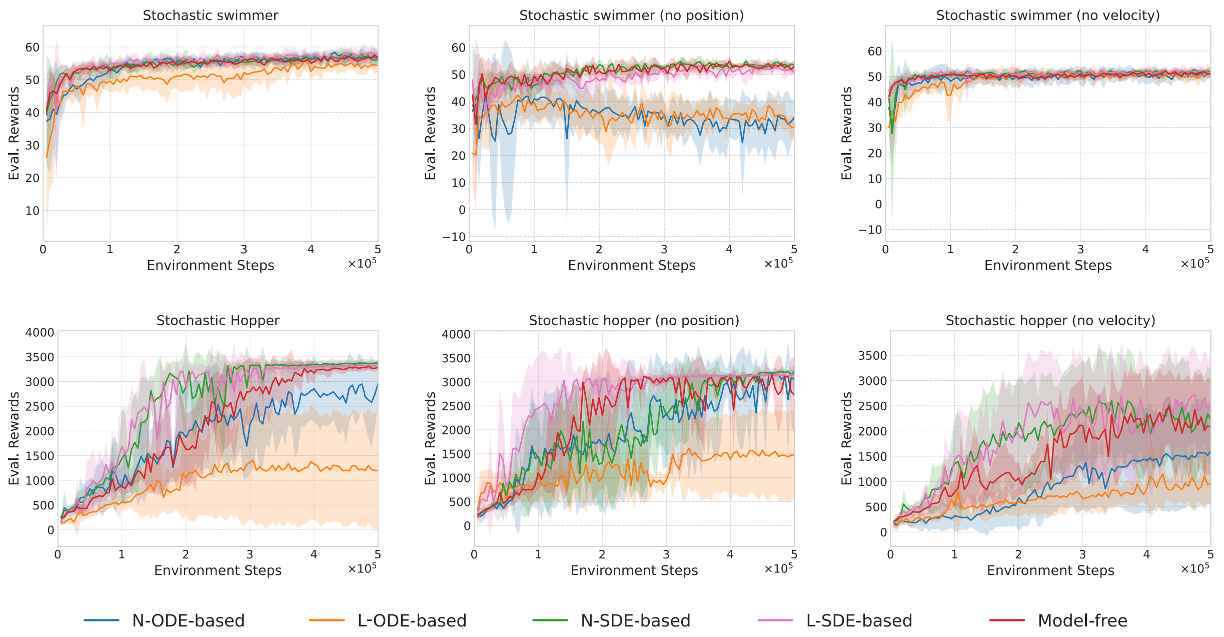

Figure 5: Learning curves of all model-based and model-free policies in the fully and partially observable environments of the stochastic swimmer and hopper. The shaded regions of each baseline show the standard deviation of the evaluated returns over 5 runs (evaluation data collected every 5k environment steps). The policies developed by the SDE-based model achieve better performance than the ODE-based policies across all stochastic environments, and the latent SDE-based model shows greater sample efficiency than the model-free baseline in the more complex hopper environment.

transition dynamics, enabling a more accurate prediction of the next (partial) observation than the ODE-based models (see Fig. 8 in the Appendix). This further enables SDE-based policies to account for transition uncertainty and potentially avoids actions associated with high risk. In contrast, ODE-based models capture only the mean dynamics, and policies planned on them may be more prone to local minima.

SDE-based policies exhibit better sample efficiency than model-free policies given enough complexity of the environment. In the stochastic Hopper environment, (see Fig. 5 bottom row), the policies with SDE-based planning demonstrate their advantage over the model-free baseline by requiring fewer interactions with the real environment (environment steps) to converge. This advantage is not observed in the simpler Swimmer environments. In particular, the model using latent SDE is the best performing in the POMDP of stochastic Hopper (no position). We further observe that the next partial observation predicted by the SDE transition in the latent space (latent SDE) aligns better with the real partial observations than those predicted by the SDE transition directly in the observation space (neural SDE) (see Fig. 9 in the Appendix), which might explain the performance advantage.

5 Conclusion

We have compared neural ODE and SDE models for model-based reinforcement learning with planning in stochastic environments, under both full and partial observability. Our findings demonstrate that policy adaptation via inverse dynamics—using state-space ODE or SDE models—is substantially more sample-efficient than learning policies from scratch when environment configurations change. Overall, SDE-based models yield better policies than their ODE counterparts, and in the most challenging partially observed settings, our proposed latent SDE model provides an advantage in sample efficiency over model-free baselines.

References

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pp. 214–223. PMLR, 2017.

Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *Aaai*, volume 94, pp. 1023–1028, 1994.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Achille Djeumou, Lingfan Huang, Alex Buisson, et al. Physics-constrained neural SDEs for efficient RL & control. In *Conference on Robot Learning (CoRL)*, 2023.

Jianzhun Du, Joseph Futoma, and Finale Doshi-Velez. Model-based reinforcement learning for semi-markov decision processes with neural odes. *Advances in Neural Information Processing Systems*, 33:19805–19816, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv [cs.LG]*, January 2018. URL `http://arxiv.org/abs/1801.01290`.

Chao Han, Debabrota Basu, Michael Mangan, Eleni Vasilaki, and Aditya Gilra. Dynamical-vae-based hindsight to learn the causal dynamics of factored-pomdps. *arXiv preprint arXiv:2411.07832*, 2024.

Patrick Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. *Advances in neural information processing systems*, 33:6696–6707, 2020.

Patrick Kidger, James Foster, Xuechen Li, Harald Oberhauser, and Terry Lyons. Neural SDEs as infinite-dimensional GANs. *arXiv [cs.LG]*, February 2021. URL `http://arxiv.org/abs/2102.03657`.

Diederik P Kingma. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Mehmet Koprulu, Aditya Dhariwal, Surya Ganguli, and Sanjeev Arora. Nuno: Neural uncertainty-aware neural SDE for offline model-based RL. In *International Conference on Learning Representations (ICLR)*, 2025.

Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3870–3882. PMLR, 2020.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *arXiv [cs.LG]*, August 2017. URL `http://arxiv.org/abs/1708.02596`.

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging state and history representations: Understanding self-predictive rl. *arXiv preprint arXiv:2401.08898*, 2024.

Yulia Rubanova, Ricky T Q Chen, and D Duvenaud. Latent ODEs for irregularly-sampled time series. *ArXiv*, abs/1907.03907, July 2019. ISSN 2331-8422. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/42a6845a557bef704ad8ac9cb4461d43-Paper.pdf`.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Liqun Zhao, Keyan Miao, Hongpeng Cao, Konstantinos Gatsis, and Antonis Papachristodoulou. NLBAC: A neural ODE-based algorithm for state-wise stable and safe reinforcement learning. *Neurocomputing*, 638: 130041, 2025.

Xinyu Zhao, Tianyu Ma, Ting Liu, Ziyan Wang, and Haoran Li. Ode-based recurrent reinforcement learning for partial observability. In *NeurIPS*, 2023.

A Stochastic cartpole environment

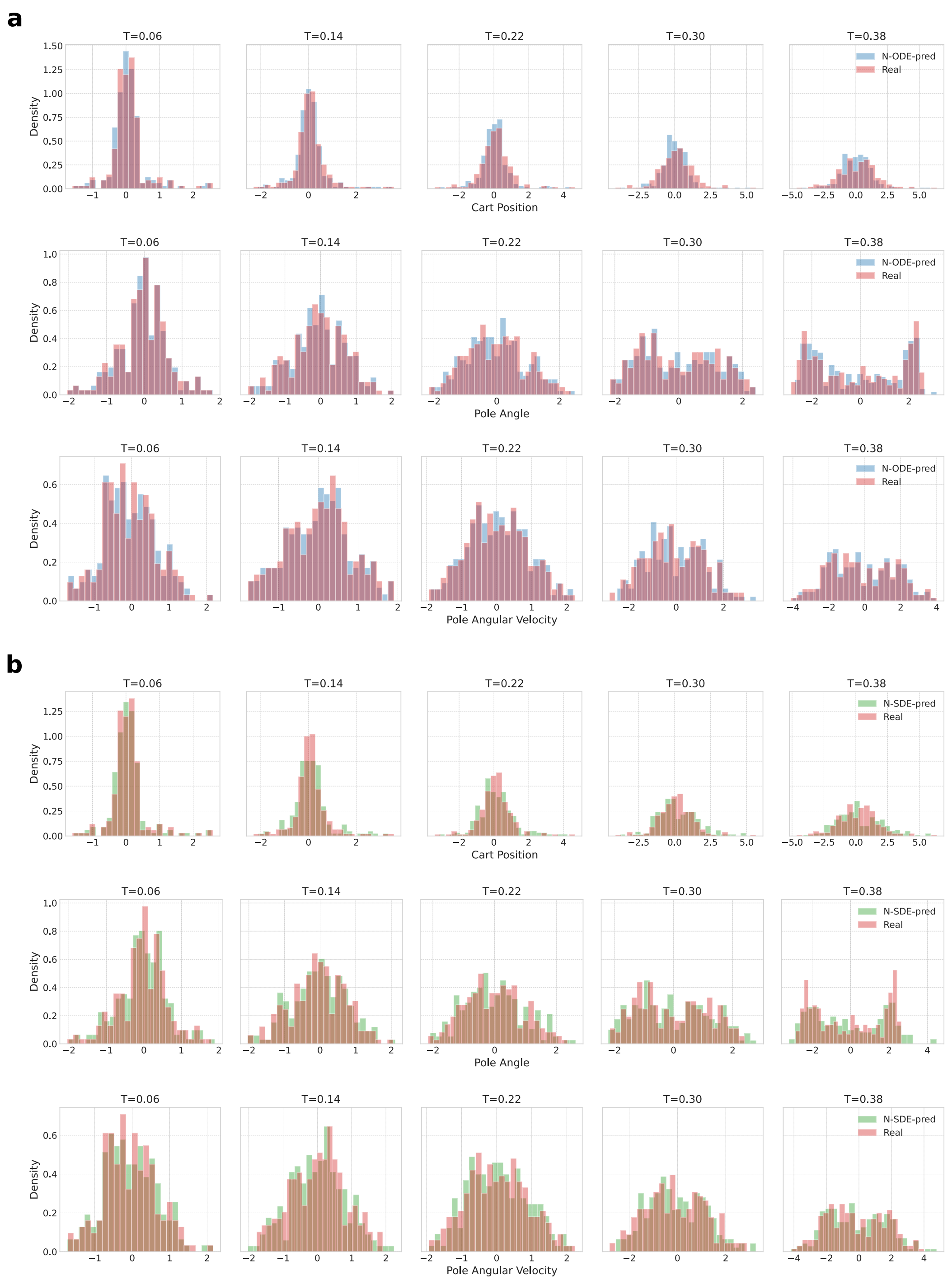

Figure 6: **(a)** Histograms of cart position, pole angle and pole angular velocity at 16%, 37%, 58%, 79%, 100% of the timesteps predicted by the neural ODE against the real histograms. **(b)** Similarly, histograms predicted by neural SDE against the real histograms.

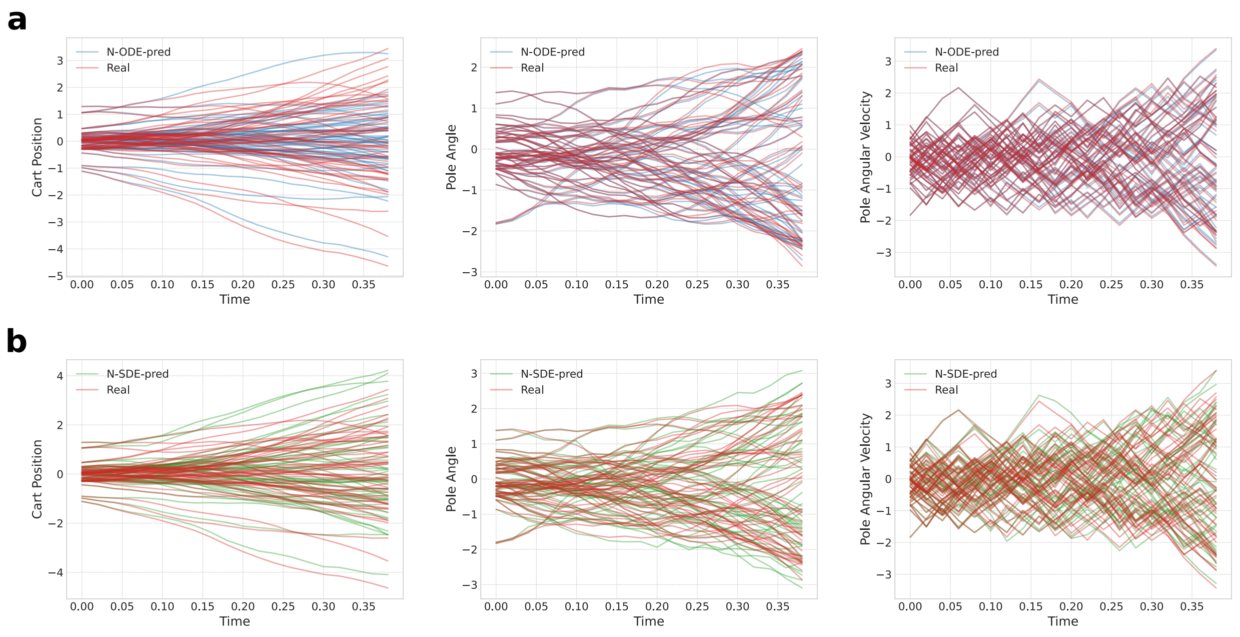

Figure 7: **(a)** 50 sample paths of cart position, pole angle and pole angular velocity predicted by neural ODE against 50 paths from the real distributions. **(b)** Similarly, paths predicted by neural SDE against paths from the real distributions. Since the transition dynamics of these features are deterministic, the neural ODE predicts almost the same distributions and their sample paths as the neural SDE does.

B Stochastic hopper environments

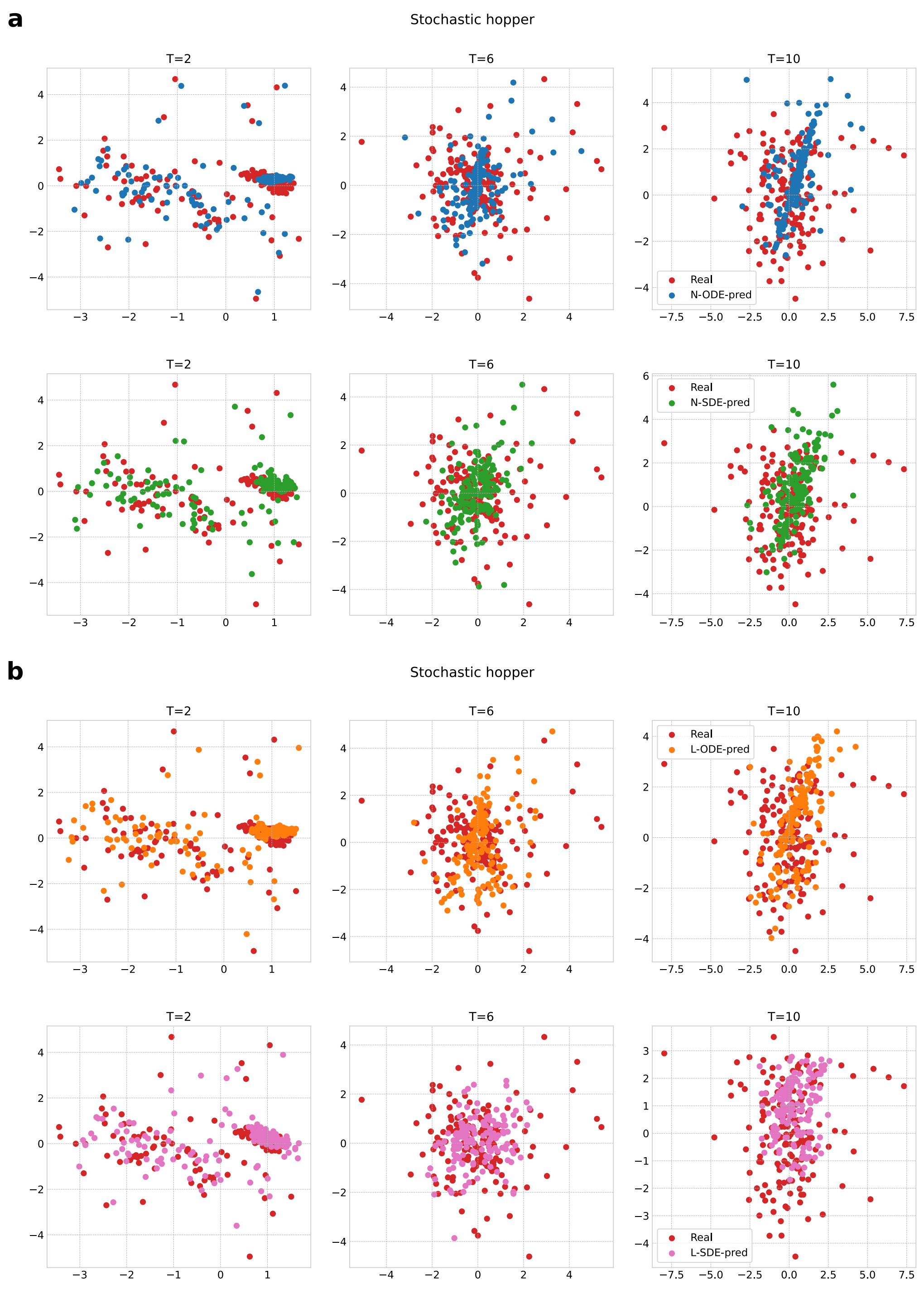

Figure 8: **(a)** Scatter plots of 200 PCA-embedded 2D representation of observations predicted by neural ODE and SDE, compared with that of the real observations across 20%, 60%, 100% of the timesteps in the stochastic hopper. **(b)** Similarly, scatter plots of 2D features predicted by latent ODE and SDE, against real ones. Predicted features by the SDE-based models demonstrate a better alignment with the real ones than those by the ODE-based models.

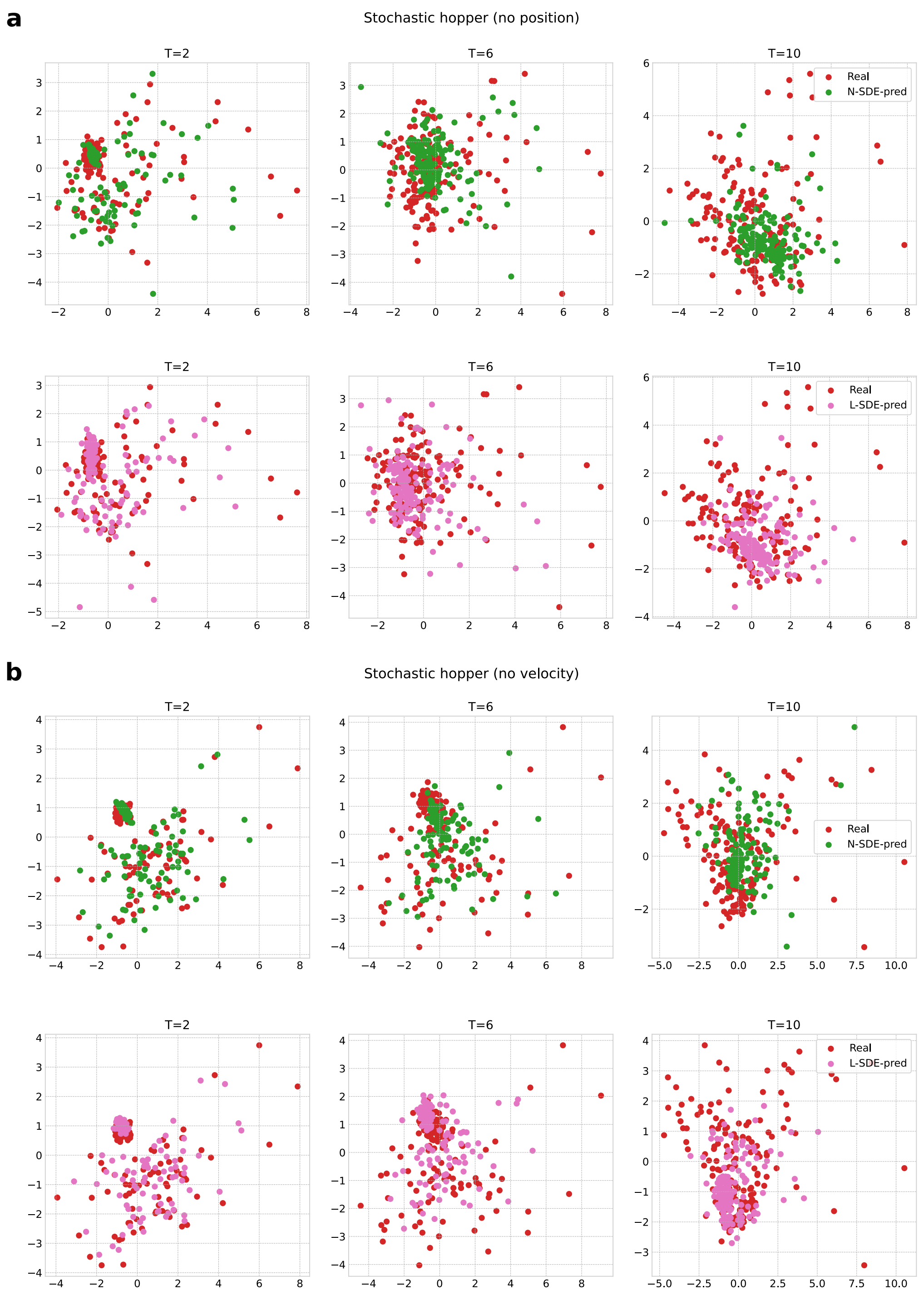

Figure 9: **(a)** Scatter plots of 200 PCA-embedded 2D representation of partial observations predicted by neural SDE and latent SDE, compared with that of the real partial observations, across 20%, 60%, 100% of the timesteps in the stochastic hopper without position observations. **(b)** Similar scatter plots in the stochastic hopper without velocity observations. Features predicted by the latent SDE align more closely with the real ones than those by the neural SDE.