Context-aware Dynamics Model
for Generalization in Model-Based Reinforcement Learning

Kimin Lee [1] [*] **Younggyo Seo** [2] [*] **Seunghyun Lee** [2] **Honglak Lee** [3] [4] **Jinwoo Shin** [2]

Abstract

Model-based reinforcement learning (RL) enjoys several benefits, such as data-efficiency and planning, by learning a model of the environments dynamics. However, learning a global model that can generalize across different dynamics is a challenging task. To tackle this problem, we decompose the task of learning a global dynamics model into two stages: (a) learning a context latent vector that captures the local dynamics, then (b) predicting the next state conditioned on it. In order to encode dynamics-specific information into the context latent vector, we introduce a novel loss function that encourages the context latent vector to be useful for predicting both forward and backward dynamics. The proposed method achieves superior generalization ability across various simulated robotics and control tasks, compared to existing RL schemes.

1. Introduction

Model-based reinforcement learning (RL) with high-capacity function approximators, such as deep neural networks (DNNs), has been used to solve a variety of sequential decision-making problems, including board games (e.g., Go and Chess (Schrittwieser et al., 2019)), video games (e.g., Atari games (Kaiser et al., 2020; Schrittwieser et al., 2019)), and complex robotic control tasks (Zhang et al., 2019; Nagabandi et al., 2019a; Hafner et al., 2020). By learning a model of the environment's dynamics and utilizing the dynamics model for planning, model-based RL achieves superior data-efficiency to model-free RL methods in general (Deisenroth & Rasmussen, 2011; Levine & Abbeel, 2014), and sometimes even beats model-free RL methods trained on sufficient amount of data (Schrittwieser

et al., 2019; Hafner et al., 2019; 2020; Kaiser et al., 2020).

However, it has been evidenced that model-based RL methods often struggle to generalize to an unseen environment with different transition dynamics (Nagabandi et al., 2019a;b). For example, when using a vanilla model-based RL method for the simple CartPole task (see Figure 1), even a minimal change in the pole mass leads to an inaccurate next-step prediction. This, in turn, leads to poor planning, and eventually performance degradation, as demonstrated in our experiments (see Table 1). Such failure to take account of transition dynamics shift makes them unreliable for real-life deployment, which requires robustness to a myriad of changing environmental factors.

To improve the generalization capabilities of model-based RL methods, several strategies have been proposed, including meta-learning (Nagabandi et al., 2019a;b) and graph networks (Sanchez-Gonzalez et al., 2018) (see Section 2 for further details). A notable such example is the model-based meta-RL method suggested by Nagabandi et al. (2019a), where a meta-learned prior model adapts to a recent trajectory segment, either by updating hidden state of a recurrent model (Duan et al., 2016), or by updating model parameters via a small number of gradient steps (Finn et al., 2017). However, it is questionable whether a simple gradient or hidden state updates would capture the rich contextual information that the environment offers. Instead, we argue that separating context encoding (i.e., capturing the contextual information) and transition inference (i.e., predicting the next state conditioned on the captured information) can be more effective for learning the environment dynamics.

Contribution. In this paper, we develop a context-aware dynamics model (CaDM) capable of generalizing across a distribution of environments with varying transition dynamics. First, to capture the contextual information, we introduce a context encoder that produces a latent vector from a recent experience. Then, by conditioning our forward dynamics model on this latent vector, we effectively perform an online adaptation to the unseen environment. We emphasize that the proposed CaDM can incorporate any dynamics model, e.g., fully-connected network or recurrent neural network, simply by conditioning the dynamics model on the encoder output. The novel ingredient of CaDM is its

loss function that forces the latent vector to be useful for predicting not only the next states (forward dynamics), but also the previous states (backward dynamics). Moreover, we consider an additional regularization that encourages temporal consistency of the context latent vector: we force the context latent vector obtained in the current timestep to be useful for predictions made in the nearby future timesteps. We discover that the proposed context latent vector aptly captures the rich contextual information of the environment, which enables fast adaptation to the changing dynamics (see Figure 6(c)). Finally, we explore the possibility of improving the generalization abilities of a model-free RL method (Schulman et al., 2017) by giving the learned context latent vector as an additional input to the policy.

We demonstrate the effectiveness of our method on a variety of simulated control tasks from OpenAI gym (Brockman et al., 2016) and MuJoCo physics engine (Todorov et al., 2012). For evaluation, we measure the performance of model-based RL methods in a range of unseen (yet related) environments with different transition dynamics (see Figure 1). In our experiments, CaDM significantly reduces the performance gap between training and test environments when compared to baselines, including ensemble methods (Chua et al., 2018) and model-based meta-RL methods (Nagabandi et al., 2019a). Furthermore, we show that CaDM can also improve model-free RL methods: it improves the generalization performance of proximal policy optimization (PPO; Schulman et al. 2017), compared to other context learning methods (Rakelly et al., 2019; Zhou et al., 2019) for model-free RL. We believe that our approach could be influential to other relevant topics, such as sim-to-real transfer (Peng et al., 2018).

2. Related Work

Model-based reinforcement learning. In model-based RL, we learn a dynamics model that approximates the true environment dynamics. Such a dynamics model can then be used for control by planning (Atkeson & Santamaria, 1997; Lenz et al., 2015; Finn & Levine, 2017), or for improving the data-efficiency of model-free RL methods (Sutton, 1990; Gu et al., 2016; Janner et al., 2019). Recently, by incorporating high-capacity function approximators like DNNs, model-based RL has shown remarkable progress even in complex domains (Oh et al., 2015; Chua et al., 2018; Hafner et al., 2019; 2020; Schrittwieser et al., 2019). For example, Chua et al. (2018) showed that an ensemble of probabilistic DNNs could reduce modeling errors and achieve the asymptotic performance on-par with model-free RL algorithms. It also has been observed that model-based RL methods can be applied for solving high-dimensional visual domain tasks (Hafner et al., 2019; 2020; Schrittwieser et al., 2019). However, developing a generalizable model-based RL method
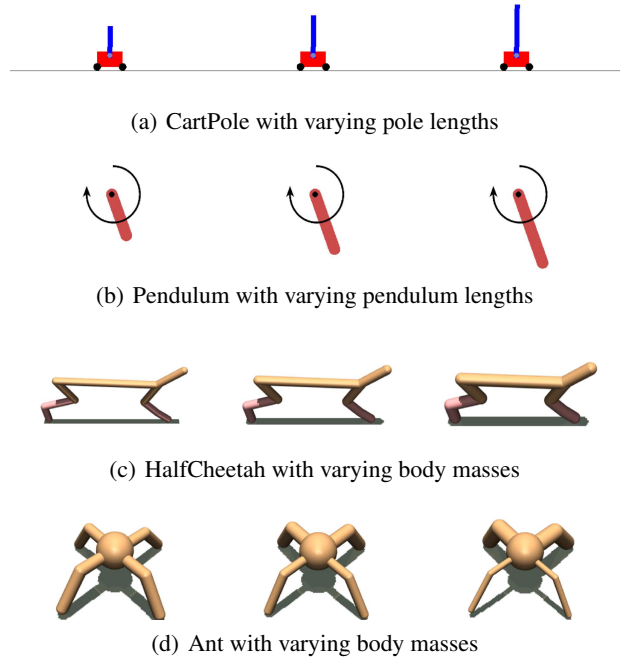

(a) CartPole with varying pole lengths

(b) Pendulum with varying pendulum lengths

(c) HalfCheetah with varying body masses

(d) Ant with varying body masses

Figure 1. Examples from (a) CartPole, (b) Pendulum, (c) HalfCheetah, and (d) Ant. We change the transition dynamics of each environment by modifying its environment parameters.

still remains a longstanding challenge.

Dynamics generalization in deep RL. Generalization of model-based RL has recently gained considerable attention in the research community. Sanchez-Gonzalez et al. (2018) proposed to model a dynamics model using a graph network, and Nagabandi et al. (2019a;b) studied model-based meta-RL methods where the meta-learner learns to update the model according to dynamics changes. Notably, Nagabandi et al. (2019a) proposed to adapt the dynamics model to recent trajectory segments, either by updating hidden representations of a recurrent model (Duan et al., 2016) or by updating model parameters via a small number of gradient steps (Finn et al., 2017). However, it is questionable whether such methods capture the environment context properly, for they are overburdened with two tasks: adaptation and next-state inference. We instead propose to disentangle the two tasks by introducing a context encoder that specializes in the former, and a conditioned dynamics model that specializes in the latter.

Several model-free RL methods, including adversarial policy training (Morimoto & Doya, 2001; Pinto et al., 2017; Rajeswaran et al., 2017), structured policy with graph neural network (Wang et al., 2018), online system identification (Yu et al., 2018; Zhou et al., 2019), and meta-learning (Finn et al., 2017; Rakelly et al., 2019), have been proposed to improve the generalization ability of RL agents across dynamics changes. In particular, Rakelly et al. (2019) pro-
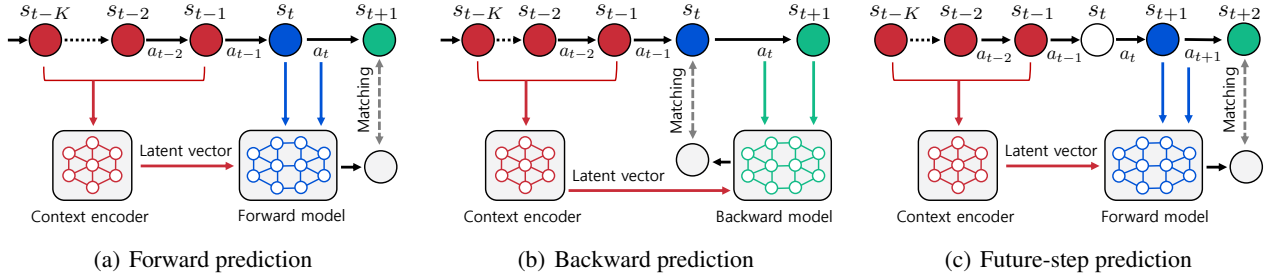
Figure 2. Illustrations of our framework. We decompose the task of learning a global dynamics model into context encoding and transition inference. (a) Our dynamics model predicts the next state conditioned on the latent vector. (b) We introduce a backward dynamics model that predicts a previous state by utilizing a context latent vector. (c) We force the context latent vector to be temporally consistent by utilizing it for predictions in the future timesteps.

posed a meta-RL method that adapts to a new environment by inferring latent context variables from a small number of trajectories. However, our method differs from this approach, in that we train our context encoder by utilizing it for dynamics prediction, as opposed to maximizing the expected returns.

3. Problem Statement

We consider the standard RL framework where an agent interacts with its environment in discrete time. Formally, we formulate our problem as a Markov Decision Process (MDP; Sutton & Barto 2018) defined as a tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0)$, with $\mathcal{S}$ denoting the state space, $\mathcal{A}$ the action space, $p\left(s'|s, a\right)$ the transition dynamic, $r\left(s, a\right)$ the reward function, ρ_0 the initial state distribution, and $\gamma \in [0, 1)$ the discount factor. The goal of RL is to find a policy that produces an action to take from each given state so as to maximize the expected return defined as the total accumulated reward. We tackle this problem in the context of model-based RL by learning a forward dynamics model f, which approximates the transition dynamics $p\left(s'|s, a\right)$.

In order to address the problem of generalization, we further consider the distribution of MDPs, where the transition dynamics $p_c\left(s'|s, a\right)$ varies according to a context c. For instance, a robot agent's transition dynamics may change when some of its parts malfunction due to unexpected damages. Our goal is to learn a generalizable forward dynamics model that is robust to such dynamics changes, i.e., approximating a distribution of transition dynamics. Specifically, given a set of training environments with contexts sampled from $p_{\mathtt{train}}(c)$, we aim to learn a forward dynamics model that can produce accurate predictions for test environments with unseen contexts sampled from $p_{\mathtt{test}}(c)$.

4. Context-aware Dynamics Model

In this section, we propose a context-aware dynamics model (CaDM) that can generalize to unseen environments with varying transition dynamics. Our scheme separates the task of reasoning about the environment dynamics into (a) encoding the dynamics-specific information into a latent vector (context encoding), and (b) predicting the next state conditioned on the latent vector (transition inference). To extract contextual information effectively, we propose a novel loss function that encourages the context latent vector to be useful for various auxiliary prediction tasks. We also discuss how the learned context latent vector can help improve the generalization abilities of model-free RL methods.

4.1. Context Encoder and Dynamics Model

To capture the true unknown context c of the environment, we introduce a context encoder g parameterized by ϕ, which produces a latent vector $z_t = g\left(\tau_{t,K}^{\mathtt{P}}; \phi\right)$ given K past transitions $\tau_{t,K}^{\mathtt{P}} = \{(s_{t-K}, a_{t-K}) \cdots (s_{t-1}, a_{t-1})\}$. The intuition is that the true context of the underlying MDP can be captured from recent experiences. Note that similar ideas have been explored by Nagabandi et al. (2019a); Rakelly et al. (2019); Zhou et al. (2019).

We introduce two dynamics models: (a) forward dynamics model $f\left(s_{t+1}|s_t, a_t, z_t; \theta\right)$ that predicts the *next state* given the current state, current action and current context latent vector (see Figure 2(a)), and (b) backward dynamics model $b\left(s_t|s_{t+1}, a_t, z_t; \psi\right)$ that predicts the *current state* given the next state, current action and current context latent vector (see Figure 2(b)). Under the assumption that MDPs with similar contexts will behave similarly (Modi et al., 2018), our method can generalize to nearby unseen dynamics by capturing context information from training environments. Forward and backward dynamics models are parameterized by θ and ψ, respectively, and are architecture-agnostic, i.e., any existing model architecture can be used. Indeed, we

experimentally confirm that the performance of stochastic models (Chua et al., 2018) significantly improves when combined with our method (see Section 5.2).

We optimize the context encoder and the dynamics models by minimizing the following loss function:

$$\mathcal{L}^{\texttt{pred}} = \mathbb{E}_{(\tau_{t,M}^{\text{F}}, \tau_{t,K}^{\text{P}}) \sim \mathcal{B}} \left[\mathcal{L}_{\texttt{forward}}^{\texttt{pred}} + \beta \mathcal{L}_{\texttt{backward}}^{\texttt{pred}} \right], \qquad (1)$$

$$\mathcal{L}_{\texttt{forward}}^{\texttt{pred}} = -\frac{1}{M} \sum_{i=t}^{t+M-1} \log f\left(s_{i+1}|s_i, a_i, g\left(\tau_{t,K}^{\text{P}}; \phi\right); \theta\right),$$

$$\mathcal{L}_{\texttt{backward}}^{\texttt{pred}} = -\frac{1}{M} \sum_{i=t}^{t+M-1} \log b\left(s_i|s_{i+1}, a_i, g\left(\tau_{t,K}^{\text{P}}; \phi\right); \psi\right),$$

where $\tau_{t,M}^{\text{F}} = \{(s_t, a_t), \cdots, (s_{t+M}, a_{t+M})\}$ is the future trajectory segment, $\mathcal{B} = \{(\tau_{t,M}^{\text{F}}, \tau_{t,K}^{\text{P}})\}$ the training dataset, $\beta > 0$ the penalty parameter, and $M > 0$ the number of future samples. Here, we remark that both of our dynamics models share the same context latent vector $g\left(\tau_{t,K}^{\text{P}}; \phi\right)$ as an additional input. Our intuition is that dynamics-specific context (e.g., environment parameters) must be useful for predicting both forward and backward transitions. Here, our motivation is that predicting backward transitions can also capture contextual information while mitigating the risk of overly focusing on predicting only the "seen" forward dynamics. Additionally, in order to handle long-horizon tasks, we encourage the context latent vector to be useful for predictions multiple timesteps into the future (see Figure 2(c)). Overall, the various prediction tasks introduced (i.e., future-step forward and backward predictions) help the context latent vector to be temporally consistent.

Using a model-based RL method to identify the environment dynamics was also studied by Zhou et al. (2019). However, the dynamics model in their method is not for producing accurate predictions, and their main focus is improving the generalizability of model-free RL methods. On the other hand, we focus on learning an accurate and generalizable dynamics model.

4.2. Additional Training Details

History of transitions. In our experiments, we use the state difference $\Delta s_t = s_{t+1} - s_t$ instead of the raw state s_t as an input to the context encoder:

$$\tau_{t,K}^{\text{P}} = \{(\Delta s_{t-K}, a_{t-K}), \cdots, (\Delta s_{t-1}, a_{t-1})\}.$$

We found that this simple technique provides further performance improvement.

Planning Algorithm. We use a model predictive control (MPC; Garcia et al. 1989) to select actions based on the forward dynamics prediction. Specifically, we use the cross entropy method (CEM; Botev et al. 2013), where N candidate action sequences are iteratively sampled from a candidate

---

Algorithm 1 Training context-aware dynamics model

1: **Inputs**: the number of past observations K and future observations M, learning rate α, and batch size B.

2: Initialize parameters of forward dynamics model θ, backward dynamics model ψ, context encoder ϕ.

3: Initialize dataset $\mathcal{B} \leftarrow \emptyset$.

4: **for** each iteration **do**

5:     // COLLECT TRAINING SAMPLES

6:     Sample $c \sim p_{\texttt{seen}}(c)$.

7:     **for** $t = 1$ **to** TaskHorizon **do**

8:         Get context latent vector $z_t = g\left(\tau_{t,K}^{\text{P}}; \phi\right)$.

9:         Collect samples $\{(s_t, a_t, s_{t+1}, r_t, \tau_{t,K}^{\text{P}})\}$ from the environment with transition dynamics p_c using the planning algorithm described in Section 4.2.

10:         Update $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, s_{t+1}, r_t, \tau_{t,K}^{\text{P}})\}$.

11:     **end for**

12:     // UPDATE DYNAMICS MODELS AND ENCODER

13:     **for** $i = 1$ **to** B **do**

14:         Sample $\tau_{i,K}^{\text{P}}, \tau_{i,M}^{\text{F}} \sim \mathcal{B}$.

15:         Get context latent vector $z_i = g\left(\tau_{i,K}^{\text{P}}; \phi\right)$.

16:         $\mathcal{L}_i^{\texttt{pred}} \leftarrow \mathcal{L}^{\texttt{pred}}\left(\tau_{i,M}^{\text{F}}, z_i, \theta, \psi\right)$ in (1).

17:     **end for**

18:     Update $\theta \leftarrow \theta - \alpha \nabla_\theta \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}_i^{\texttt{pred}}$

19:     Update $\psi \leftarrow \psi - \alpha \nabla_\psi \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}_i^{\texttt{pred}}$

20:     Update $\phi \leftarrow \phi - \alpha \nabla_\phi \frac{1}{B} \sum_{i=1}^{B} \mathcal{L}_i^{\texttt{pred}}$.

21: **end for**

---

distribution, which is adjusted based on best performing action samples. Then, we use the mean of adjusted candidate distribution as action and re-plan at every timestep.

4.3. Combination with Model-free RL

As a natural extension, we show that the learned context latent vector can be utilized for improving the generalization abilities of model-free RL methods. Here, we briefly mention that various model-free RL methods are also known to suffer from poor generalization (Yu et al., 2018; Packer et al., 2018; Zhou et al., 2019). One of the major research directions for dynamics generalization in model-free RL is developing a context-conditional policy. Specifically, Yu et al. (2018); Rakelly et al. (2019); Zhou et al. (2019) showed that a policy can be more robust to dynamics changes when it takes context information as an additional input. Motivated by this, we consider a context-conditioned policy $\pi\left(a_t|s_t, g\left(\tau_{t,K}^{\text{P}}; \phi\right)\right)$ conditioned on our learned context latent vector. Compared to existing context-conditional policies (Rakelly et al., 2019; Zhou et al., 2019), our model-free RL method demonstrates superior generalization abilities (see Table 2).

5. Experiments

In this section, we evaluate the performance of our CaDM method to answer the following questions:

- Is CaDM more robust to dynamics changes compared to other model-based RL methods (see Table 1)?

- Can CaDM be combined with model-free RL methods to improve their generalization abilities (see Table 2)?

- Does the proposed prediction loss (1) improve the test performance (see Figure 6(a))?

- Can CaDM make accurate predictions (see Figure 6(b) and Figure 7)?

- Does our context encoder extract meaningful contextual information (see Figure 6(c))?

5.1. Setups

Environments. We demonstrate the effectiveness of our proposed method on simulated robots (i.e., HalfCheetah, Ant, CrippledHalfCheetah, and SlimHumanoid) using the MuJoCo physics engine (Todorov et al., 2012) and classic control tasks (i.e., CartPole and Pendulum) from OpenAI Gym (Brockman et al., 2016). The goal of CartPole is to prevent the pole from falling over by pushing the cart left and right, while that of Pendulum is to swing up the pendulum and keep it in the upright position. As for HalfCheetah, Ant, and SlimHumanoid, the goal is to move forward as fast as possible, while keeping the control cost minimal. As in Packer et al. (2018); Zhou et al. (2019), we modify the environment parameters (e.g., mass, length, damping) that characterize the transition dynamics (see Figure 1). In the case of CrippledHalfCheetah, one of the actuators is randomly crippled to change the transition dynamics.

For both training and testing, we sample environment parameters at the beginning of each episode. During training, we randomly sample environment parameters from a predefined training range. At test time, we measure each model's performance in unseen environments characterized by parameters outside the training range. In order to utilize model-predictive control (MPC), we assume that the reward function is known, as in (Chua et al., 2018; Nagabandi et al., 2019a).

Note that generalization performance is measured in two different regimes: moderate and extreme, where the former draw environment parameters from a closer range to the training range, compared to the latter (see Figure 3). For all our experiments, we select the model with the highest average return during training and report the test performance. We report mean and standard deviation across five runs. Due to space limitations, we provide more experimental details in the supplementary material.
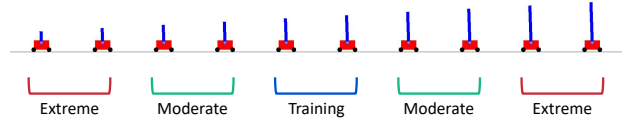

Figure 3. Illustration of training and test parameter ranges.

Baselines and our method. We consider the following model-based RL methods as baselines:

- Vanilla dynamics model (Vanilla DM): Dynamics model trained to minimize the standard one-step forward prediction loss. The model is fixed during test time (i.e., no adaptation).

- Stacked dynamics model (Stacked DM): Vanilla dynamics model which takes the past $K \in \{5, 10, 15\}$ transitions as an additional input. A comparison with this model evaluates the benefit of introducing a context latent vector.

- Gradient-Based Adaptive Learner (GrBAL; Nagabandi et al. 2019a): Model-based meta-RL method which trains a dynamics model by optimizing an adaptation meta-objective. At test time, the meta-learned prior dynamics model adapts to a recent trajectory segment by taking gradient steps.[1]

- Recurrence-Based Adaptive Learner (ReBAL; Nagabandi et al. 2019a): Model-based meta-RL method similar to GrBAL. However, instead of taking gradient steps, ReBAL uses a recurrent model that learns its own update rule, i.e. updating its hidden state.[1]

- Probabilistic ensemble dynamics model (PE-TS; Chua et al. 2018): An ensemble of probabilistic dynamics models designed to incorporate both environment stochasticity and subjective uncertainty into the model.

We combine CaDM with two baseline model-based RL methods, Vanilla DM and PE-TS.: the results of ReBAL + CaDM can also be found in the supplementary material, where it underperforms Vanilla + CaDM or PE-TS + CaDM overall. We remark that the proposed method can be applied to any model-based RL methods because we separate a context encoder from forward dynamics models, as shown in Figure 2. Due to space limitation, we provide details about model architecture and hyperparameters in the supplementary material. Our code is available at https://github.com/younggyoseo/CaDM.

---

[1] We used a reference implementation publicly available at https://github.com/iclavera/learning_to_adapt.

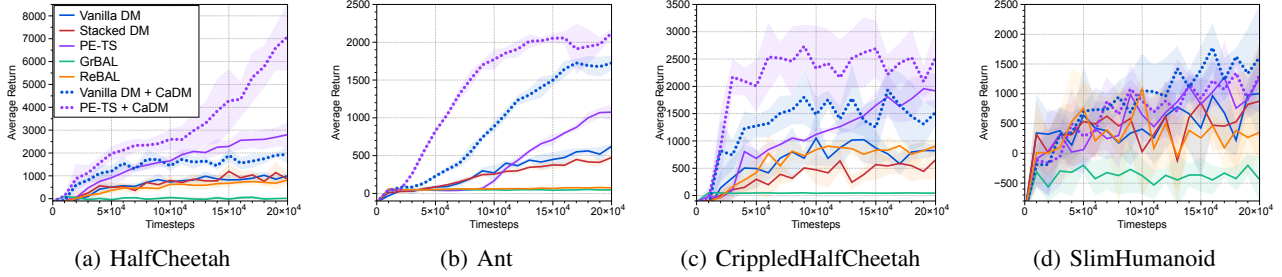(a) HalfCheetah (b) Ant (c) CrippledHalfCheetah (d) SlimHumanoid

Figure 4. The average returns of trained dynamics models on unseen (moderate) environments. The results show the mean and standard deviation of returns averaged over five runs. The full figures of all environments are in the supplementary material.

	CartPole			Pendulum		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
Vanilla DM	$176.6\pm_{10.2}$	$124.7\pm_{8.1}$	$105.9\pm_{3.7}$	$-419.2\pm_{95.1}$	$-928.4\pm_{56.8}$	$-1170.6\pm_{51.3}$
Stacked DM	$176.9\pm_{6.3}$	$131.9\pm_{6.0}$	$106.6\pm_{11.0}$	$-1158.7\pm_{124.7}$	$-1308.0\pm_{49.6}$	$-1288.3\pm_{65.3}$
GrBAL	$118.1\pm_{40.2}$	$87.5\pm_{35.1}$	$81.6\pm_{21.3}$	$-726.8\pm_{322.6}$	$-1027.7\pm_{68.1}$	$-1048.1\pm_{111.8}$
ReBAL	$105.7\pm_{45.8}$	$48.4\pm_{34.5}$	$54.3\pm_{37.0}$	$-460.4\pm_{74.5}$	$-860.8\pm_{140.3}$	$-1026.4\pm_{51.5}$
PE-TS	$194.3\pm_{4.1}$	$171.3\pm_{18.5}$	$142.9\pm_{23.6}$	$-577.2\pm_{198.1}$	$-985.7\pm_{64.2}$	$-1221.1\pm_{31.0}$
Vanilla + CaDM	$185.7\pm_{7.7}$	$154.3\pm_{13.2}$	$118.6\pm_{6.6}$	$\mathbf{-415.2}\pm_{60.4}$	$\mathbf{-593.7}\pm_{48.6}$	$\mathbf{-967.9}\pm_{83.5}$
PE-TS + CaDM	$\mathbf{196.1}\pm_{4.8}$	$\mathbf{187.3}\pm_{11.2}$	$\mathbf{149.4}\pm_{26.2}$	$-537.0\pm_{114.6}$	$-705.5\pm_{41.7}$	$-1098.4\pm_{47.1}$

	Half-cheetah			Ant		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
Vanilla DM	$1560.7\pm_{453.1}$	$1026.7\pm_{164.7}$	$686.7\pm_{189.4}$	$646.4\pm_{89.0}$	$520.0\pm_{97.6}$	$385.8\pm_{85.2}$
Stacked DM	$1301.4\pm_{310.5}$	$761.1\pm_{236.6}$	$661.5\pm_{220.5}$	$492.3\pm_{68.7}$	$417.1\pm_{46.8}$	$338.9\pm_{51.5}$
GrBAL	$117.0\pm_{88.7}$	$-43.7\pm_{106.9}$	$-94.5\pm_{141.3}$	$55.0\pm_{10.0}$	$46.5\pm_{6.5}$	$42.9\pm_{3.8}$
ReBAL	$1086.7\pm_{90.0}$	$657.5\pm_{184.9}$	$396.6\pm_{188.5}$	$100.1\pm_{12.3}$	$73.1\pm_{15.5}$	$53.0\pm_{17.2}$
PE-TS	$4347.1\pm_{300.9}$	$2019.6\pm_{274.8}$	$1422.3\pm_{162.8}$	$1183.3\pm_{51.1}$	$1075.1\pm_{103.6}$	$856.6\pm_{66.5}$
Vanilla + CaDM	$3536.5\pm_{641.7}$	$1556.1\pm_{260.6}$	$1264.5\pm_{228.7}$	$1851.0\pm_{113.7}$	$1315.7\pm_{45.5}$	$821.4\pm_{113.5}$
PE-TS + CaDM	$\mathbf{8264.0}\pm_{1374.0}$	$\mathbf{7087.2}\pm_{1495.6}$	$\mathbf{4661.8}\pm_{783.9}$	$\mathbf{2848.4}\pm_{61.9}$	$\mathbf{2121.0}\pm_{60.4}$	$\mathbf{1200.7}\pm_{21.8}$

	CrippledHalfCheetah			SlimHumanoid		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
Vanilla DM	$1005.1\pm_{429.0}$	$870.0\pm_{308.0}$	$577.3\pm_{76.5}$	$1119.8\pm_{317.6}$	$1004.4\pm_{798.2}$	$1155.5\pm_{556.9}$
Stacked DM	$630.6\pm_{211.3}$	$545.1\pm_{289.8}$	$417.9\pm_{145.8}$	$1057.4\pm_{547.5}$	$876.2\pm_{1005.2}$	$651.8\pm_{449.9}$
GrBAL	$151.9\pm_{122.7}$	$-9.2\pm_{17.1}$	$16.6\pm_{23.0}$	$-62.6\pm_{233.1}$	$-562.8\pm_{253.5}$	$-398.6\pm_{177.2}$
ReBAL	$701.7\pm_{119.7}$	$904.5\pm_{90.7}$	$833.0\pm_{118.0}$	$1205.8\pm_{546.8}$	$85.8\pm_{388.9}$	$108.7\pm_{357.6}$
PE-TS	$1846.8\pm_{380.7}$	$1916.5\pm_{328.2}$	$1227.6\pm_{35.2}$	$1339.6\pm_{524.0}$	$758.6\pm_{528.8}$	$810.4\pm_{363.4}$
Vanilla + CaDM	$2435.1\pm_{880.4}$	$1375.3\pm_{290.6}$	$966.9\pm_{89.4}$	$\mathbf{1758.2}\pm_{459.1}$	$\mathbf{1228.9}\pm_{374.0}$	$\mathbf{1487.9}\pm_{339.0}$
PE-TS + CaDM	$\mathbf{3294.9}\pm_{733.9}$	$\mathbf{2618.7}\pm_{647.1}$	$\mathbf{1294.2}\pm_{214.9}$	$1371.9\pm_{400.0}$	$903.7\pm_{343.9}$	$814.5\pm_{274.8}$

Table 1. The performance (average returns) of trained dynamics models on various control tasks. The transition dynamics of environments are changing in both training and test environments. The results show the mean and standard deviation of returns averaged over five runs.

5.2. Comparison with the Model-based RL Methods

Table 1 shows the performance of various model-based RL methods on both training and test environments (see the supplementary material for training curve plots). Our method significantly improves both training and test performances of various model-based methods in all the environments. Especially, the performance gain due to CaDM becomes much more significant in more complex environments (e.g., long-horizon and high-dimensional domains like halfChee-

tah, Ant, and Humanoid. For example, when combined with PE-TS, CaDM improves the average return from 2019.6 to 7087.2 for the HalfCheetah environment in the moderate regime. This demonstrates the applicability of our method to any model-based RL methods. One important result is that stacking input transitions sometimes degrades the performances in both training and test environments, which implies that our approach to separate context encoding and transition inference is indeed more effective for approximating true context c than a naïve stacking method. We also
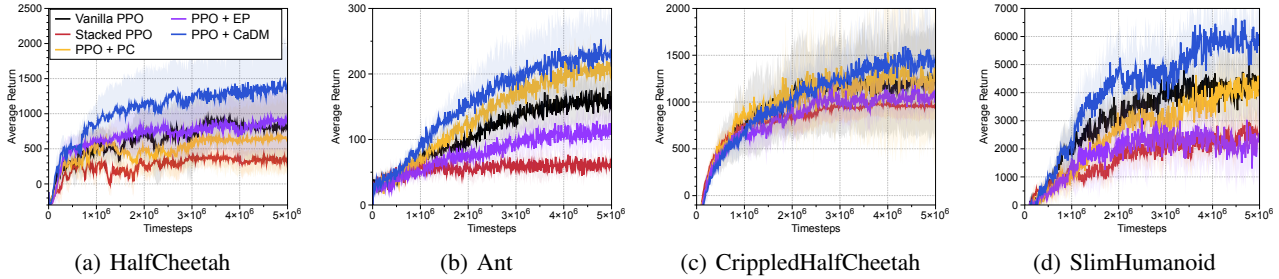
Figure 5. The average returns of trained agents on unseen (moderate) environments. The results show the mean and standard deviation of returns averaged over five runs. The full figures of all environments are in the supplementary material.

	CartPole			Pendulum		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
Vanilla PPO	**200.0**± 0.0	**199.1**± 0.9	187.8± 4.7	-945.1± 126.0	-1113.2± 69.1	-1356.8± 48.0
Stacked PPO	198.6± 1.7	197.8± 1.3	189.2± 6.1	-316.9± 197.3	-475.7± 228.1	-488.2± 178.2
PPO + PC	**200.0**± 0.0	198.0± 1.4	187.5± 10.9	-451.1± 248.6	-645.3± 320.7	-1136.6± 251.0
PPO + EP	**200.0**± 0.0	196.3± 4.0	184.5± 9.7	-255.7± 9.5	-374.3± 24.6	**-256.7**± 26.4
PPO + CaDM	**200.0**± 0.0	197.9± 3.0	**193.0**± 3.5	**-199.3**± 22.2	**-279.8**± 42.1	-426.4± 227.0

	HalfCheetah			Ant		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
Vanilla PPO	2043.4± 802.9	807.7± 553.6	574.0± 645.6	211.9± 44.5	149.4± 27.0	117.3± 23.1
Stacked PPO	1125.4± 85.5	361.1± 141.7	5.7± 208.1	90.6± 16.3	53.2± 10.6	46.0± 10.9
PPO + PC	1584.9± 404.3	642.1± 488.3	462.1± 534.5	249.9± 85.0	207.0± 33.8	163.5± 30.4
PPO + EP	1620.9± 491.5	895.3± 445.1	674.2± 686.8	138.8± 34.9	107.8± 19.9	93.5± 32.4
PPO + CaDM	**2652.0**± 1133.6	**1224.2**± 630.0	**1021.1**± 676.6	**268.6**± 77.0	**228.8**± 48.4	**199.2**± 52.1

	CrippledHalfCheetah			SlimHumanoid		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
Vanilla PPO	2059.6± 658.3	1223.6± 559.9	781.7± 270.3	7685.5± 2599.4	3761.3± 1582.4	2751.6± 869.4
Stacked PPO	1238.1± 102.5	967.1± 146.6	904.4± 146.5	4831.0± 688.1	2443.0± 535.6	1577.8± 573.5
PPO + PC	**2920.7**± 771.7	1162.2± 456.5	546.3± 215.9	7130.1± 3378.0	3928.5± 1848.7	2362.6± 781.9
PPO + EP	1494.2± 311.7	1017.0± 201.1	719.0± 438.5	4824.7± 1508.7	2224.7± 882.9	1293.4± 729.0
PPO + CaDM	2356.6± 624.3	**1454.0**± 462.6	**1025.0**± 296.2	**10455.0**± 1004.9	**4975.7**± 1305.7	**3015.1**± 1508.3

Table 2. The performance (average return) of trained agents on various control tasks. The transition dynamics of environments are changing in both training and test environments. The results show the mean and standard deviation of returns averaged over five runs.

found that the proposed method is more stable compared to model-based meta-RL methods (i.e., GrBAL and ReBAL), which update model parameters or hidden states.

5.3. Comparison with the Model-free RL Methods

We also verify whether the learned context latent vector is useful for improving the generalization performance of model-free RL methods. Specifically, similar to Henderson et al. (2018), we use MLPs with two hidden layers of 64 units and tanh activations for the policy network, and the Proximal Policy Optimization (PPO; Schulman et al. 2017) method to train the agents. Our proposed method, which takes the learned context latent vector (denoted PPO + CaDM), is compared with several context-conditional policies (Rakelly et al., 2019; Zhou et al., 2019). Specif-

ically, we consider the stacked PPO, which takes the past $K \in \{5, 10, 15\}$ transitions as an additional input, and PPO with probabilistic context (PPO + PC), which learns context variable by maximizing the expected returns (Rakelly et al., 2019). We also consider PPO with environment probing policy (PPO + EP) that takes embeddings extracted from initial interaction with an environment as an additional input (Zhou et al., 2019). Due to space limitations, we provide more detailed explanations in the supplementary material.

Table 2 shows the performance of various model-free RL methods on both training and test environments (see the supplementary for training curve plots). In complex environments, such as HalfCheetah, Ant, CrippledHalfCheetah, and SlimHumanoid, our PPO + CaDM shows better generalization performances than previous conditional policy
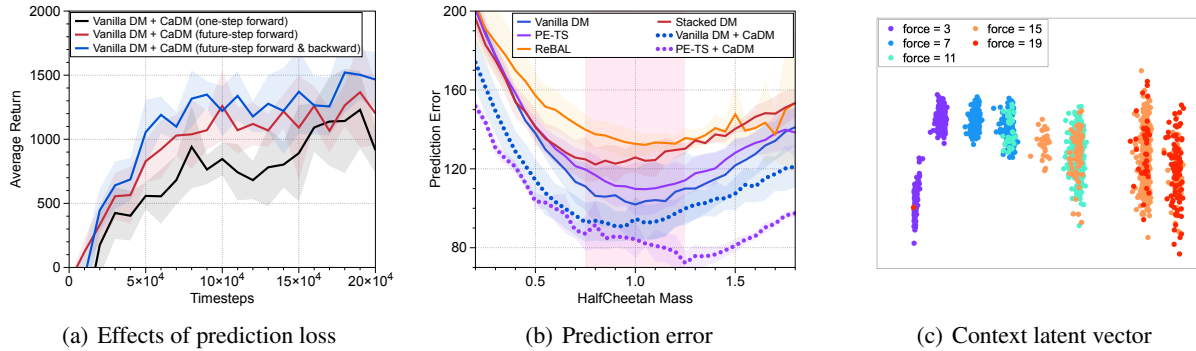
(a) Effects of prediction loss (b) Prediction error (c) Context latent vector

Figure 6. (a) Test performance of dynamics models optimized by variants of the proposed prediction objective in (1) on HalfCheetah environments. (b) Prediction errors on the HalfCheetah task with varying mass values. The pink shaded area represents the training range. (c) PCA visualization of context latent vectors extracted from trajectories collected in the CartPole environments. Embedded points from environments with the same mass parameter have the same color. The full figures of all environments are in the supplementary material.
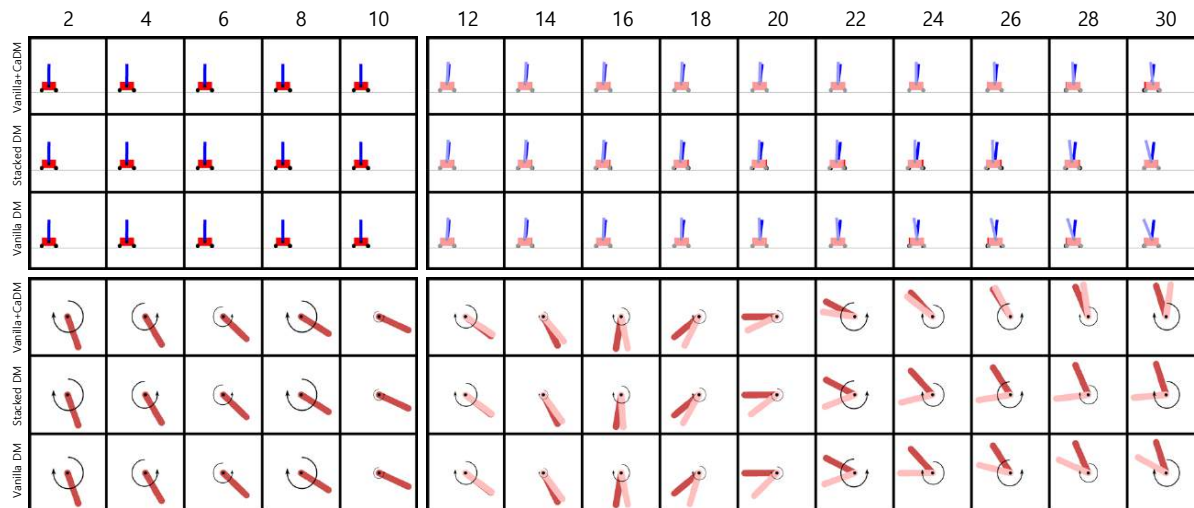
Figure 7. Visualization of future state predictions from Vanilla + CaDM (ours), Stacked DM, and Vanilla DM on CartPole and Pendulum environments with unseen environment parameters. Given 10 past states and actions, we predict future states for the next 20 timesteps only using ground truth actions. The colored and faded objects represent the ground truth states and the predicted states, respectively.

methods (i.e., PPO + EP and PPO + PC), implying that the proposed CaDM method can extract contextual information more effectively. On the other hand, the performance gain from our method is marginal in simple environments, such as CartPole and Pendulum.

5.4. Ablation Study

Effects of prediction loss. In order to verify the individual effects of the suggested prediction losses (1), we train three different dynamics models that optimize (a) one-step forward prediction loss, (b) future-step forward prediction loss, and (c) future-step forward and backward prediction loss, respectively. Figure 6(a) shows the test performances for the HalfCheetah environment in the extreme regime. One can observe that optimizing the future-step forward and backward prediction loss achieves the best performances. This shows that every component in the proposed loss function

helps the context encoder extract the environment context.

Prediction errors. To show that our method indeed helps with a forward prediction, we compare baseline methods with CaDM in terms of prediction error across multiple HalfCheetah environments with varying mass values. As shown in Figure 6(b), our model demonstrates superior prediction performance in both training and test environments. In particular, PE-TS quickly becomes unreliable outside the training range (pink shaded area), whereas our model maintains a tolerable level of prediction errors throughout.

Embedding analysis. We analyze whether the learned latent vector encodes meaningful information about the environment. To this end, we collect trajectories from CartPole environments with different push force magnitudes, then visualize the latent encoding of the collected trajectory segments using principal component analysis (PCA) (Jolliffe, 1986). As shown in Figure 6(c), latent vectors

extracted from environments with different mass parameters are clearly separated in the embedding space. In contrast, raw state vectors are scattered and disjointed (see Figure 21(a) in the supplementary material), which implies that our context encoder captures useful contextual information.

Prediction visualization. We also visualize the future state predictions in test environments from CartPole and Pendulum in Figure 7 with unseen environment parameters (i.e., force and mass). Given 10 past states and actions, we generate 20 future state predictions from Vanilla DM, Stacked DM, and Vanilla + CaDM only using ground truth actions. One can observe that Vanilla + CaDM gives much more accurate predictions compared to other models. Notably, Vanilla DM and Stacked DM fail to provide accurate predictions for more distant future timesteps, while CaDM consistently gives accurate predictions across all timesteps, which confirms our belief that the proposed method can capture contextual information about the transition dynamics.

6. Conclusion

In this paper, we propose a context-aware dynamics model that adapts to dynamics changes. To learn a generalizable dynamics model, we separate context encoding (i.e., capturing the contextual information) and transition inference (i.e., predicting the next state conditioned on the captured information). By forcing the context latent vector to be useful for both forward dynamics and backward dynamics, our method aptly captures the contextual information. We also showed that the learned context vector can be utilized to improve the generalization performance of model-free methods. We believe our work can serve as a strong guideline in related topics.

Acknowledgements

References

Atkeson, C. G. and Santamaria, J. C. A comparison of direct and model-based reinforcement learning. In *ICRA*, 1997.

Botev, Z. I., Kroese, D. P., Rubinstein, R. Y., and LEcuyer, P. The cross-entropy method for optimization. In *Handbook of statistics*. Elsevier, 2013.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 2011.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Finn, C. and Levine, S. Deep visual foresight for planning robot motion. In *ICRA*, 2017.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Garcia, C. E., Prett, D. M., and Morari, M. Model predictive control: theory and practicea survey. *Automatica*, 25(3): 335–348, 1989.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. Continuous deep q-learning with model-based acceleration. In *ICML*, 2016.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In *ICML*, 2019.

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In *AAAI*, 2018.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *NeurIPS*, 2019.

Jolliffe, I. T. Principal components in regression analysis. In *Principal component analysis*, pp. 129–155. Springer, 1986.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., et al. Model-based reinforcement learning for atari. In *ICLR*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Lenz, I., Knepper, R. A., and Saxena, A. Deepmpc: Learning deep latent features for model predictive control. In *RSS*, 2015.

Levine, S. and Abbeel, P. Learning neural network policies with guided policy search under unknown dynamics. In *NeurIPS*, 2014.

Maaten, L. v. d. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov): 2579–2605, 2008.

Modi, A., Jiang, N., Singh, S., and Tewari, A. Markov decision processes with continuous side information. In *ALT*, 2018.

Morimoto, J. and Doya, K. Robust reinforcement learning. In *NeurIPS*, 2001.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *ICLR*, 2019a.

Nagabandi, A., Finn, C., and Levine, S. Deep online learning via meta-learning: Continual adaptation for model-based rl. In *ICLR*, 2019b.

Oh, J., Guo, X., Lee, H., Lewis, R. L., and Singh, S. Action-conditional video prediction using deep networks in atari games. In *NeurIPS*, 2015.

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., and Song, D. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *ICRA*, 2018.

Pinto, L., Davidson, J., Sukthankar, R., and Gupta, A. Robust adversarial reinforcement learning. In *ICML*, 2017.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. Epopt: Learning robust neural network policies using model ensembles. In *ICLR*, 2017.

Rakelly, K., Zhou, A., Quillen, D., Finn, C., and Levine, S. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, 2019.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Sanchez-Gonzalez, A., Heess, N., Springenberg, J. T., Merel, J., Riedmiller, M., Hadsell, R., and Battaglia, P. Graph networks as learnable physics engines for inference and control. In *ICML*, 2018.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sutton, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, 1990.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT Press, 2018.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy with graph neural networks. In *ICLR*, 2018.

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., and Ba, J. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Yu, W., Tan, J., Liu, C. K., and Turk, G. Preparing for the unknown: Learning a universal policy with online system identification. In *RSS*, 2018.

Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M. J., and Levine, S. Solar: deep structured representations for model-based reinforcement learning. In *ICML*, 2019.

Zhou, W., Pinto, L., and Gupta, A. Environment probing interaction policies. In *ICLR*, 2019.