PROTOTYPICAL CONTEXT-AWARE DYNAMICS GENERALIZATION FOR HIGH-DIMENSIONAL MODEL-BASED REINFORCEMENT LEARNING

Junjie Wang[1,2], Yao Mu[3], Dong Li[4], Qichao Zhang[1,2], Dongbin Zhao[1,2], Yuzheng Zhuang[4], Ping Luo[3], Bin Wang[4], and Jianye Hao[4]

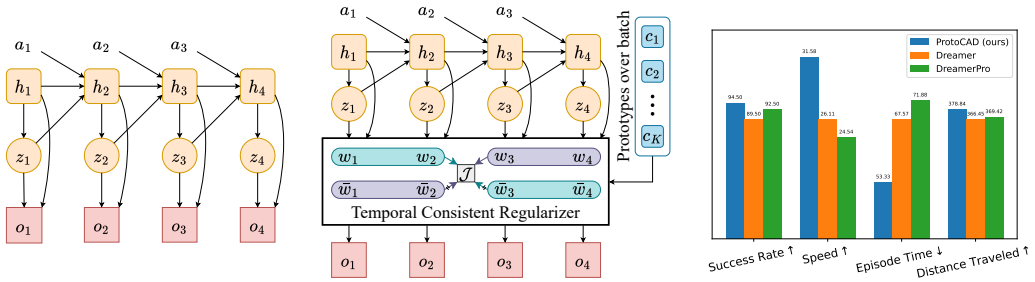
[1]State Key Laboratory of Multimodal Artificial Intelligence Systems, Institute of Automation, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]The University of Hong Kong
[4]Huawei Noah's Ark Lab

ABSTRACT

The ability to generalize different dynamics is crucial for decision-making in autonomous driving that relies on high-dimensional inputs. The latent world model provides a promising way to learn policies in a compact latent space for tasks with high-dimensional observations, however, its generalization across diverse environments with unseen dynamics remains challenging. Although the recurrent structure utilized in current advances helps to capture local dynamics, modeling only state transitions without an explicit understanding of environmental context limits the generalization ability of the dynamics model. To address this issue, we propose a **Proto**typical **C**ontext-**A**ware **D**ynamics (**ProtoCAD**) model, which captures the local dynamics by temporally consistent latent context and enables dynamics generalization in high-dimensional control tasks. ProtoCAD extracts useful contextual information with the prototypes clustered over the batch, and it benefits model-based reinforcement learning in two ways: 1) A temporally consistent prototypes regularizer is utilized, which encourages the prototype assignments produced for different temporal parts of the same latent trajectory to be temporally consistent instead of comparing the features; 2) A context representation is designed, which combines both projection embedding of latent states and aggregated prototypes and can significantly improve the dynamics generalization ability. Extensive experiments show that ProtoCAD surpasses existing methods in terms of dynamics generalization.

1 INTRODUCTION

Autonomous driving with high-dimensional observations requires a powerful characterizer to make sense of the surrounding environment and generalize to new situations. In case autonomous driving decision-making is modeled as a Reinforcement Learning (RL) problem, different traffic flow densities, driving styles, road segments, weather, etc. can lead to different dynamics in the Markov Decision Process (MDP). Thus, RL policies with dynamics generalization capability are critical for autonomous driving. Latent world models (Ha & Schmidhuber, 2018) summarize an agent's experience from high-dimensional observations to facilitate learning complex behaviors in a compact latent space. Current advances (Hafner et al., 2019; 2020; Wang et al., 2022) leverage Recurrent Neural Networks (RNNs) to extract historical information from high-dimensional observations as compact latent representations and enable imagination in the latent space. It is shown that RNN-based models that model only latent state transitions have a certain ability to generalize across different dynamics (Lu et al., 2022). Recently, some works of dynamics generalization on low-dimensional tasks (Lee et al., 2020; Seo et al., 2020; Guo et al., 2022) demonstrate that extracting environmental context information from historical trajectories as additional input to the model can benefit both model learning and policy planning, and can improve the generalization ability among different dynamics. An intuitive question is "*How to extract effective environmental context information under*



(a) RSSM (b) ProtoCAD (ours) (c) Test performance on CARLA

Figure 1: Comparison of different latent world models. We introduce a novel temporally consistent prototypical regularizer into the RSSM (Hafner et al., 2019) framework. Here, h and z denote deterministic and stochastic states, respectively, and o denotes observation. The temporal crossover self-supervised loss provided by this additional regularizer facilitates the extraction of dynamics-related states by the model, while the learned prototypes summarize the characteristics of the seen environments in the experience of the agent. Compared to Dreamer (Hafner et al., 2020), which uses RSSM as a dynamics model, and DreamerPro (Deng et al., 2022), which combines RSSM and prototypes for representation, ProtoCAD has better generalization performance.

high-dimensional observations if it helps to improve the dynamics generalization of RNN-based latent world models?" Unfortunately, in high-dimensional observation space, it is difficult to simply apply the aforementioned methods to directly derive environmental context information and rollout the dynamics model for policy planning. For tasks with high-dimensional sensor inputs, dynamics generalization remains challenging. Therefore, in this paper, we investigate how to equip the latent world model with context information about the environment to cope with dynamics generalization.

To reduce the difficulty of extracting contextual information from high-dimensional observations, we present the **Proto**typical **C**ontext-**A**ware **D**ynamics (**ProtoCAD**) model, which learns context using data-clustering prototypes. The prototypes summarize the characteristics of dynamics in historical experience and are flexible to extend to unseen dynamics. Instead of comparing features directly, we enforce temporal consistency between prototype assignments produced for different time parts of the same observation sequence. Specifically, we first make two different augmentations from the historical observation sequence and embed them into latent states. Then, those latent states are fed into linear projectors to obtain the projection embedding. To extract accurate dynamics-specific information, we regularize the projections and prototypes to be time consistent in a sequence and invariant to spatial perturbations by a modified temporal crossover SwAV (Caron et al., 2020) loss. By calculating the probability that the projections are matched with prototypes, we can obtain an aggregate prototype by combining the learned prototypes with the probabilities as weights. Finally, both the projection embedding and aggregated prototypes are combined as the context representation for policy learning to capture contextual information among different dynamics. Figure 1 illustrates a brief schematic of ProtoCAD compared to the Recurrent State-Space Model (RSSM), a dynamics model commonly adopted for high-dimensional input tasks. Works related to this paper are listed in Appendix A.1. To the best of our knowledge, this is the first approach that addresses the dynamics generalization problem of high-dimensional inputs. The contributions of this work are listed below.

- We propose ProtoCAD, a Model-Based Reinforcement Learning (MBRL) framework that brings a temporally consistent prototypical regularizer to the latent world model. The latent model can learn more efficient representations for dynamics generalization thanks to the backpropagation gradient provided by this additional structure and the designed temporal crossover SwAV loss.
- We design a novel context representation that incorporates projection embedding and an aggregated prototype based on the predicted probabilities. The effectiveness of this representation is verified by combining it with the latent state as a complete context-based latent feature.
- We develop various environments with different transition dynamics to evaluate the performance of ProtoCAD, including discrete autonomous driving decision-making with high-dimensional observations, continuous visual control, and state-input control. Extensive ex-

periments demonstrate that our approach achieves better zero-shot dynamics generalization performance.

2 PROBLEM STATEMENT AND PRELIMINARIES

2.1 PROBLEM STATEMENT

Formally, we formulate the problem of high-dimensional autonomous driving as a discrete-time Partially Observable Markov Decision Process (POMDP), since the underlying state of the environment cannot be obtained directly from the high-dimensional sensory input. A POMDP is a 7-tuple $\mathcal{M} \doteq (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, P, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, and $\mathcal{O}$ is the set of observations. $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the conditional transition probability that action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ will lead to state s_{t+1}. $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ denotes the reward function and $P : \mathcal{S} \times \mathcal{A} \to \mathcal{O}$ denotes the observation probabilities. $\gamma \in (0, 1)$ is a discount factor. The definitions of the observation, action, and reward function for the autonomous driving task are given in Appendix A.4.1.

Consider a context set $\mathcal{C}$, where different $c \in \mathcal{C}$ result in different POMDP models. For example, the mass of a pendulum can be considered contextual information. Same as in previous works (Lee et al., 2020; Seo et al., 2020; Guo et al., 2022), we assume that a context does not change within an episode, only between episodes, and the distribution of contexts is uniform across the context set. For the dynamics generalization problem, the entire set of contexts can be divided into two subsets: $\mathcal{C}_{\text{train}}$ and $\mathcal{C}_{\text{test}}$. In this paper, we focus on zero-shot dynamics generalization, i.e., $\mathcal{C}_{\text{train}} \cap \mathcal{C}_{\text{test}} = \emptyset$. Given a context c, we can compute the expected return of policy π with $G(\pi, \mathcal{M}|_c) \doteq \mathbb{E}_\pi \left(\sum_{t=0}^{\infty} \gamma^t r_t\right)$, where $\mathcal{M}|_c$ is the POMDP conditioned on c. And for any POMDP M, we can define the expected return of policy π with $G(\pi, \mathcal{M}) \doteq \mathbb{E}_{c \sim p(c)} \left[G(\pi, \mathcal{M}|_c)\right]$. The goal is to find a policy trained on $\mathcal{C}_{\text{train}}$ that maximizes the expected return on the testing context set: $\mathcal{J}(\pi) \doteq G(\pi, \mathcal{M}|_{\mathcal{C}_{\text{test}}})$.

2.2 PRELIMINARIES

Many MBRL approaches first learn a world model and then further exploit it to derive policies. Typically, the world model provides a mapping of environmental dynamics from the current state and action to the next state. To extract compact representations from image observation sequences, RSSM (Hafner et al., 2019) separates states into stochastic and deterministic components, allowing the model to robustly learn to predict multiple futures. RSSM is commonly made up of the following components (Hafner et al., 2019; 2021),

$$\begin{aligned} &\text{Recurrent module:} && h_t = f_\phi\left(h_{t-1}, z_{t-1}, a_{t-1}\right) \\ &\text{Representation module:} && z_t \sim q_\phi\left(z_t \mid h_t, o_t\right) \\ &\text{Transition module:} && \hat{z}_t \sim p_\phi\left(\hat{z}_t \mid h_t\right), \end{aligned} \tag{1}$$

where o_t is the current observation at time step t, h_t denotes the deterministic recurrent state, $\hat{z}_t$, as well as z_t, denote the stochastic states of the prior and posterior, respectively, and ϕ is the parameter of the model. Here, we denote the deterministic output by f, the distribution of samples generated in the real environment by q, and their approximation by p. The optimization objective of the model is to reduce the KL distance between the prior and the posterior,

$$\mathcal{J}_{\text{RSSM}}^t \doteq -\beta \text{KL}\left[q_\phi\left(z_t \mid h_t, o_t\right) \parallel p_\phi\left(\hat{z}_t \mid h_t\right)\right], \tag{2}$$

where β is a hyperparameter controlling the loss scale.

3 METHOD

In this section, we present the model-based reinforcement learning framework ProtoCAD. In order to learn a context-aware world model that facilitates dynamics generalization and policy training, the entire process of learning ProtoCAD is divided into three parts, including latent state encoding, prototypical context learning, and policy optimization. Figure 2 provides an overview of the learning process of the prototypical context-aware dynamics model. First, latent state encoding: the raw observations are augmented to obtain two different views, and the two augmented historical trajectories are encoded as latent space states through a transition model RSSM; second, prototypical

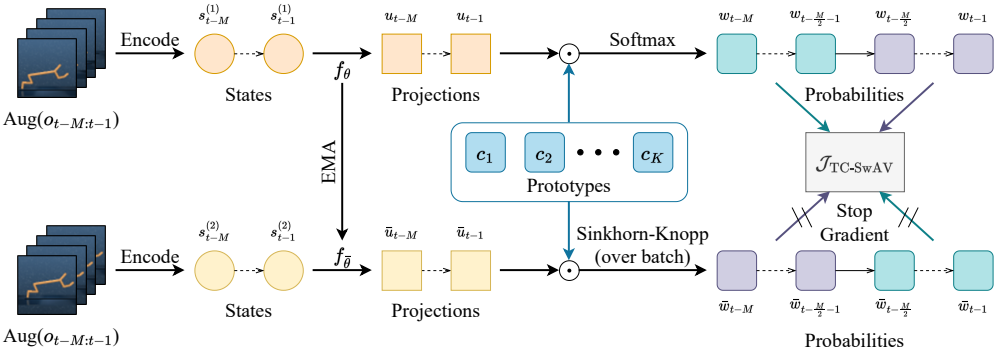


Figure 2: The learning process of prototypical context-aware dynamics model (ProtoCAD). ProtoCAD aims to extract efficient contextual information from high-dimensional observation sequences with the help of prototypes clustered over batch, which summarize the characteristics of dynamics in historical experience. Spatially, different augmentations from the same observation sequence can be dynamics consistent when the augmentations are ensured to be consistent over time steps. Temporally, different segments of the same trajectory share the same dynamics. Thus, we first make two augmentations from the original observation sequences. Then, the augmented observation sequences are encoded into latent states via RSSM. Instead of comparing features directly, we enforce temporal consistency between prototype assignments produced for different time parts of the same latent trajectory. Both the projector f_θ and the prototypes $\{c_k\}_{k=1}^K$ are updated by the temporal crossover loss $\mathcal{J}_{\text{TC-SwAV}}$.

context learning: linear projections are implemented on the two latent space states to predict cluster assignments, and we enforce temporal consistency between prototype assignments produced for different time parts of the same latent trajectory. Then we aggregate the prototypes with the probabilities, and the projections are matched with the prototypes as weights. Finally, the projections and aggregated prototypes are combined as contextual features as a condition for policy optimization. The pseudocode of our overall algorithm is shown in Appendix A.2.

3.1 Latent State Encoding

In this paper, we implement RSSM as the transition model for partially observable environments. During the training process, a sequence of historical observations $o_{t-M:t-1}$ and actions $a_{t-M:t-1}$ are sampled from the experience replay buffer, and we first augment the observations to two different views $o^{(1)}_{t-M:t-1}$ and $o^{(2)}_{t-M:t-1}$ with data augmentation. Following DreamerPro (Deng et al., 2022), we perform random shifts with bilinear interpolation and ensure consistency of the augmentation across time steps. We assume that this augmentation does not change the dynamics information of the sequences. Subsequently, augmented observations, together with an action sequence, are fed into the RSSM to obtain the latent states $s^{(i)}_\tau \doteq (h^{(i)}_\tau, z^{(i)}_\tau)$, $\tau = t-M, t-M+1, \cdots, t-1$, $i \in \{1, 2\}$.

The deterministic recurrent structure in the transition model combined with stochastic state inference allows it to encode states that both remember multi-step historical information and include the ability to capture environmental uncertainty. This mechanism enables the model to have some generalization capability. However, a simple gradient for latent state updates is insufficient to capture the rich contextual information provided by the environment (Lee et al., 2020), which in turn limits the ability of the model to generalize and transfer policies based on it. Therefore, there is an emerging demand for context-aware dynamics modeling.

3.2 Prototypical Context Learning

Self-supervised learning approaches show great potential to learn effective representations from high-dimensional data. SwAV (Caron et al., 2020), for example, proposes learning embedding by matching them to a set of learned clusters. Coincidentally, for the generalization problem with a finite number of dynamics settings, the contextual representations of different environments lie in some clusters. In contrast to existing algorithms (Yarats et al., 2021a; Mazoure et al., 2022; Deng

et al., 2022) that introduce SwAV into RL to help state representation, ProtoCAD groups the latent states into K sets and combines learned prototypes with projection embeddings to capture contextual information in different dynamics environments. We now introduce how to extract prototypical context representations from latent states.

The K trainable prototypes $\{c_k\}_{k=1}^K$ can be regarded as corresponding to the K potential "situations" in which RL agents may find themselves (Mazoure et al., 2022). To cluster the latent states output by RSSM into K prototypes, states $s^{(1)}_{t-M:t-1}$ are first fed into a linear projector f_θ (the deterministic output is still denoted by f) and then $\ell 2$ normalized to obtain projections $u_{t-M:t-1}$. Subsequently, a softmax operation is carried out on the dot product of $u_{t-M:t-1}$ and prototypes,

$$w_{t-M:t-1,k} = \text{softmax}\left(\frac{u_{t-M:t-1} \cdot c_k}{T}\right), k = 1, 2, \cdots, K. \tag{3}$$

where $w_{t-M:t-1,k}$ is the predicted probability that projections $u_{t-M:t-1}$ map to cluster k, T is a temperature parameter, and the prototypes $\{c_k\}_{k=1}^K$ are also $\ell 2$ normalized.

To train both the projector and prototypes, we make a copy of the projector, called the target projector ($f_{\bar{\theta}}$), whose parameters $\bar{\theta}$ are updated by Exponential Moving Average (EMA) of θ, and cluster its output to compute target projections. The Sinkhorn-Knopp (Knight, 2008) algorithm, which has been widely used for online clustering to assist RL tasks (Yarats et al., 2021a; Mazoure et al., 2022; Deng et al., 2022), is adopted because it is ideal to evolve online clustering with the arrival of new batches of trajectories. As ProtoCAD is an online operation, Sinkhorn-Knopp is better suited for this task than other clustering methods. Latent states $s^{(2)}_{t-M:t-1}$ are fed into $f_{\bar{\theta}}$ with $\ell 2$ normalization to get target projections $\bar{u}_{t-M:t-1}$, and target probabilities $\{\bar{w}_{t-M:t-1,k}\}_{k=1}^K$ are derived by applying the Sinkhorn-Knopp algorithm to $\bar{u}_{t-M:t-1}$ and $\{c_k\}_{k=1}^K$.

A fundamental setting of an agent's dynamics is that it does not change within a trajectory. Therefore, we design a temporal crossover SwAV loss to encourage the temporal consistency of the learned features, including projections and prototypes. We divide the above-obtained probabilities w and $\bar{w}$ into two parts in the time dimension (the sequence length M is set to be even in the implementation), i.e.,

$$y_{(1),k} \doteq y_{t-M:t-M/2-1,k}, y_{(2),k} \doteq y_{t-M/2:t-1,k}, y \in \{w, \bar{w}\}, \tag{4}$$

and then back-propagate the gradient by the following objective,

$$\begin{aligned} \mathcal{J}_{\text{TC-SwAV}} &\doteq \frac{1}{2}\sum_{k=1}^{K}\left(\bar{w}_{(1),k}\cdot\log w_{(2),k} + \bar{w}_{(2),k}\cdot\log w_{(1),k}\right) \\ &= \frac{1}{2}\sum_{k=1}^{K}\left(\sum_{\tau=t-M}^{t-M/2-1}\bar{w}_{\tau,k}\log w_{\tau+M/2,k} + \sum_{\tau=t-M/2}^{t-1}\bar{w}_{\tau,k}\log w_{\tau-M/2,k}\right). \end{aligned} \tag{5}$$

Here, TC stands for temporal consistency. Up to this point, we can obtain the aggregated prototypes of the latent state among the clusters by projector f_θ and prototypes $\{c_k\}_{k=1}^K$ with $e_t \doteq \sum_{k=1}^K w_{t,k} \cdot c_k$. Then, we obtain context representations including the projection embedding u_t and aggregated prototypes e_t. Combined with the latent state s_t, this yields a complete latent feature:

$$x_t \doteq (s_t, u_t, e_t). \tag{6}$$

Considering that the reward function is independent of the environment dynamics (or context), the original observation and reward are predicted from the latent feature space and the latent state space, respectively. That is,

$$\text{Image predictor:} \quad \hat{o}_t \sim p_\phi\left(\hat{o}_t \mid x_t\right), \quad \text{Reward predictor:} \quad \hat{r}_t \sim p_\phi\left(\hat{r}_t \mid s_t\right). \tag{7}$$

The distributions produced by the image predictor and reward predictor are trained to maximize the log-likelihood of their corresponding targets,

$$\mathcal{J}_{\text{O}}^t \doteq \ln p_\phi(\hat{o}_t|x_t), \quad \mathcal{J}_{\text{R}}^t \doteq \ln p_\phi(\hat{r}_t|s_t). \tag{8}$$

To sum up, the overall objective of the prototypical context-aware dynamics model learning is,

$$\mathcal{J}_{\text{ProtoCAD}} \doteq \mathbb{E}_{p_\phi}\left(\sum_t\left(\mathcal{J}_{\text{RSSM}}^t + \mathcal{J}_{\text{O}}^t + \mathcal{J}_{\text{R}}^t\right) + \mathcal{J}_{\text{TC-SwAV}}\right). \tag{9}$$

3.3 POLICY LEARNING

We implement the actor-critic architecture for behavior learning. Benefiting from the world model equipped with prototypes, imagination can be performed in the latent space with context features. For latent feature x_t, the actor and critic models are defined as,

$$\text{Actor:} \quad a_t \sim \pi_\psi(a_t|x_t), \quad \text{Critic:} \quad v_\xi(x_t) \approx \mathbb{E}_{\pi(\cdot|x_t)}\left(\sum_{\tau=t}^{t+H} \gamma^{\tau-t}\hat{r}_\tau\right). \tag{10}$$

After getting context-aware features of the future through the rollout of the world model, we can further predict the rewards and values of future states and obtain the target values to train actor and critic networks (see more details in the supplementary material). For the estimation of target values, there is a trade-off between model utilization and its prediction accuracy. As the number of model rollout steps increases, the model provides more data for policy training, resulting in higher sample efficiency. At the same time, the accuracy of model prediction decreases. Therefore, we weight the multi-step value estimates to calculate the target value, as in Dreamer,

$$\begin{aligned} V_N^i(x_\tau) &\doteq \mathbb{E}_{p_\phi,\pi_\psi}\left(\sum_{n=\tau}^{h-1}\gamma^{n-\tau}\hat{r}_n + \gamma^{h-\tau}v_\xi\left(x_h\right)\right) \text{ with } h = \min\left(\tau+i, t+H\right) \\ V_\lambda(x_t) &\doteq (1-\lambda)\sum_{n=1}^{H-1}\lambda^{n-1}V_N^n(x_t) + \lambda^{H-1}V_N^H(x_t), \end{aligned} \tag{11}$$

where $\tau = t, t+1, \cdots, t+H$. The learning objectives of the actor and critic models are set as

$$\mathcal{J}_{\text{Actor}} \doteq \mathbb{E}_{p_\phi,\pi_\psi}\left(\sum_{\tau=t}^{t+H} V_\lambda(x_\tau)\right), \quad \mathcal{J}_{\text{Critic}} \doteq -\mathbb{E}_{p_\phi,\pi_\psi}\left(\sum_{\tau=t}^{t+H}\frac{1}{2}\|v_\psi(x_\tau) - V_\lambda(x_\tau)\|^2\right). \tag{12}$$

4 EXPERIMENTS

4.1 SETUPS

We evaluate our method on a wide range of tasks with varying dynamics, including discrete visual decision-making on CARLA (Dosovitskiy et al., 2017), continuous visual control on DM-Control (DMC) (Tunyasuvunakool et al., 2020), and state-input control on MuJoCo (Todorov et al., 2012). For each task, our evaluation is taken in a zero-shot manner, i.e., we train under some set of dynamics and test under some other unseen dynamics settings. To verify the superiority of the proposed ProtoCAD model, we compare it with several state-of-the-art (SOTA) model-based and model-free methods. For tasks with high-dimensional input (CARLA and DMC), our baselines include Dreamer (Hafner et al., 2020), DreamerPro (Deng et al., 2022), and DrQ (Yarats et al., 2021b). For tasks with low-dimensional input (MuJoCo), our baselines include Dreamer, CaDM (Lee et al., 2020), and TMCL (Seo et al., 2020). The specific experimental settings are given in Appendix A.4.

4.2 EVALUATION ON GENERALIZATION ABILITY

Generalization on discrete visual autonomous driving decision-making. In CARLA, we construct the training and testing environments by controlling the density of traffic flow. After the training is completed, we test the strategies learned by the different algorithms at unseen traffic flow densities. A total of 200 episodes are tested, and the success rate (the percentage of episodes without collisions), average speed, average lane change times, average episode time, and average distance traveled (before a collision occurs) are statistically measured. The result is shown in Table 1. While the expectation is to have as few lane changes as possible, there is no optimal value for the average lane changes times metric itself. For example, DreamerPro does not learn to change lanes, which is clearly not a promising policy. In the rest of the metrics, ProtoCAD achieves the best results among different methods.

Generalization on continuous visual control. Across diverse DMC tasks, agents trained on the training parameter set by different methods are evaluated under unseen dynamics settings. The

Table 1: Results of generalization on discrete visual autonomous driving decision-making.

	DrQ	Dreamer	DreamerPro	ProtoCAD (ours)
Average lane change times	0.62	0.18	0.00	1.6
Success Rate ↑	89.5%	89.5%	92.5%	**94.5%**
Average speed (km/h) ↑	29.19	26.11	24.54	**31.58**
Average episode time (s) ↓	59.9	67.57	71.88	**53.33**
Average distance traveled (m) ↑	373.65	366.45	369.42	**378.84**

performance comparison results are illustrated in Table 2. For DreamerPro, we maintain the parameter settings of the original paper with its publicly available code. In all experimental tasks, ProtoCAD exceeds all model-based baselines with significant improvements. These experimental results suggest that combining prototypes into the latent world model can substantially improve the model's ability to generalize to different transition dynamics. Although DrQ achieves better or comparable performance compared to model-based methods among several methods in Finger Spin, its performance on Hopper Hop, Pendulum Swingup, and Quadruped Run is noticeably worse than model-based algorithms. Table 2 also demonstrates the overall performance of different methods on several experimental tasks, from which it can also be seen that the comprehensive performance of DrQ is inferior to the other methods. The mean and median performance of ProtoCAD on several experimental tasks is significantly better than the other methods. Specifically, compared to Dreamer, ProtoCAD improves the mean and median performance by 19.9% and 17.1%, respectively.

Table 2: Results of generalization on continuous visual control ($\pm$ denotes the standard deviation).

	DrQ	Dreamer	DreamerPro	ProtoCAD (ours)
Cheetah Run	$471.1_{\pm 49.0}$	$450.9_{\pm 57.5}$	$394.3_{\pm 60.2}$	$\mathbf{537.9}_{\pm 46.7}$
Finger Spin	$\mathbf{592.3}_{\pm 48.4}$	$453.0_{\pm 58.3}$	$458.5_{\pm 74.0}$	$585.0_{\pm 162.1}$
Hopper Hop	$73.7_{\pm 16.2}$	$112.6_{\pm 31.2}$	$148.6_{\pm 77.5}$	$\mathbf{176.6}_{\pm 95.2}$
Pendulum Swingup	$504.0_{\pm 272.1}$	$712.8_{\pm 82.7}$	$737.0_{\pm 49.9}$	$\mathbf{802.0}_{\pm 15.3}$
Quadruped Run	$251.0_{\pm 184.6}$	$444.1_{\pm 21.7}$	$502.6_{\pm 59.3}$	$\mathbf{529.0}_{\pm 36.5}$
Walker Run	$378.7_{\pm 50.1}$	$392.9_{\pm 31.1}$	$378.0_{\pm 34.9}$	$\mathbf{447.1}_{\pm 38.4}$
Mean (overall)	$385.8_{\pm 71.3}$	$427.7_{\pm 9.7}$	$436.4_{\pm 17.2}$	$\mathbf{512.9}_{\pm 31.1}$
Median (overall)	$421.6_{\pm 57.9}$	$436.9_{\pm 29.4}$	$423.5_{\pm 12.9}$	$\mathbf{511.6}_{\pm 40.4}$

Generalization on state-input control. Here, we evaluate ProtoCAD on state-input tasks with the same experimental environment setup as RIA (Guo et al., 2022). Since methods like CaDM and TMCL utilize planning to obtain actions and assume a known reward function on state transitions during the planning process, they are difficult to deploy in high-dimensional input tasks. Thus, we compare ProtoCAD with them in this section. The results are given in Table 3. We report the average rewards of ProtoCAD over 5 seeds (baselines are 3). In all tasks, the dynamics generalization capability of ProtoCAD is comparable to or surpasses that of CaDM and Dreamer. In the HalfCheetah task, our approach does not perform as well as TMCL. One possible reason is that the multi-head dynamics model introduced in TMCL is very effective for this task. We leave the combination of multi-head dynamics with ProtoCAD for our future work.

Table 3: Results of generalization on state-input control ($\pm$ denotes the standard deviation).

	CaDM	TMCL	Dreamer	ProtoCAD (ours)
Pendulum	$-713.95_{\pm 21.1}$	$-691.2_{\pm 93.4}$	$-575.9_{\pm 56.6}$	$\mathbf{-525.9}_{\pm 61.6}$
Ant	$1660_{\pm 57.8}$	$2994.9_{\pm 243.8}$	$4636.3_{\pm 412.8}$	$\mathbf{5309.4}_{\pm 537.7}$
Hopper	$845.2_{\pm 20.41}$	$999.35_{\pm 22.8}$	$2107.3_{\pm 36.1}$	$\mathbf{2197.8}_{\pm 83.44}$
HalfCheetah	$5876.6_{\pm 799.0}$	$\mathbf{9039.6}_{\pm 1065}$	$4701.0_{\pm 796.2}$	$5409.8_{\pm 594.1}$
C_HalfCheetah	$3656.4_{\pm 856.2}$	$3998.8_{\pm 856.2}$	$\mathbf{4445.0}_{\pm 316.4}$	$4125.4_{\pm 957.9}$
Slim_Humanoid	$859.1_{\pm 24.01}$	$2098.7_{\pm 109.1}$	$14735_{\pm 5842.9}$	$\mathbf{16731.6}_{\pm 9595.8}$

4.3 Evaluation on sample efficiency

Figure 3 illustrates the performance comparison of ProtoCAD and baselines on different zero-shot dynamics generalization continuous visual control environments. The evaluation is performed in environments with unseen dynamics. Solid lines represent the mean score, and shaded areas mark the standard deviation across 5 seeds. ProtoCAD is comparable to or better than baselines in all tasks.

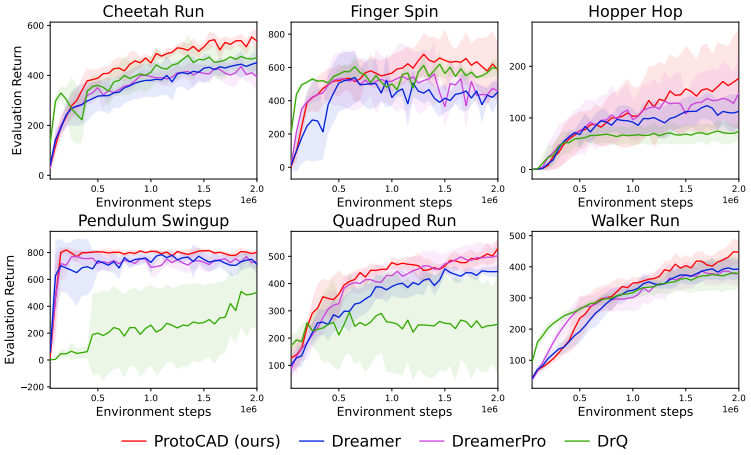


Figure 3: Evaluation on sample efficiency.

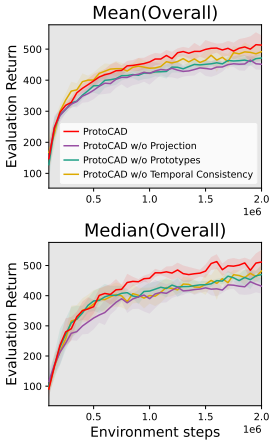


Figure 4: Ablations.

To further validate the effectiveness of ProtoCAD in tasks that do not involve multi-dynamics generalization, we compare it to DreamerV2 (Hafner et al., 2021), a SOTA model-based approach, on the standard DMC benchmark (without varying the dynamics). As shown in Figure 8 (see Appendix A.6.1), ProtoCAD also outperforms DreamerV2 in this benchmark.

4.4 Ablation Study

ProtoCAD integrates learned prototypes based on predicted probabilities, together with the projection embedding as a context representation, while using the cross-correspondence between predicted and target probabilities in time sequence to make the context features consistent over a trajectory. We investigate the contribution of each part of ProtoCAD by removing the individual component from it.

ProtoCAD without projection embedding (w/o Projection): In this setting, the context representation is derived from the combination of prototypes and the latent state, i.e., $x_t = (s_t, e_t)$.
ProtoCAD without prototypes incorporating (w/o Prototypes): In this setting, the context representation is combined from the embedding of the projector and the latent state of the RSSM, i.e. $x_t = (s_t, u_t)$.
ProtoCAD without temporal dimensional cross-loss (w/o Temporal Consistency): In this setting, the SwAV loss is calculated from a one-to-one correspondence between the predicted probability w and the target probability $\bar{w}$ in time sequence, i.e., replace $\mathcal{J}_{\text{TC-SwAV}}$ with $\mathcal{J}_{\text{SwAV}} = \frac{1}{2}\sum_{k=1}^{K}\left(\bar{w}_{(1),k}\cdot\log w_{(1),k} + \bar{w}_{(2),k}\cdot\log w_{(2),k}\right)$.

We plot the mean and median performance for all tasks in Figure 4. It can be seen that each composition contributes significantly to the performance of ProtoCAD, combining all of them achieves the best results across different tasks. In addition, we also visualize the learned features with TSNE. See Appendix A.6.2 for more results.

5 Conclusion

Dynamics generalization with high-dimensional observations is a critical yet challenging problem in autonomous driving with reinforcement learning. In this paper, we propose a novel model-based framework, **Proto**typical **C**ontext-**A**ware **D**ynamics (**ProtoCAD**) model, which introduces prototypes into the latent world model while simultaneously performing latent space representation learning and temporally consistent context clustering. Evaluations of challenging lane-changing decision-making and visual control tasks with unseen dynamics demonstrate that our approach achieves state-of-the-art performance. Our ablation experiments further illustrate that the superiority of ProtoCAD is attributed to the context representation that combines projections and prototypes as well as the temporal consistency loss. In addition, extending our framework by incorporating advanced task-relevant information extraction techniques to further improve the dynamics generalization capability could be left as our future work.

REFERENCES

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*, pp. 1597–1607. PMLR, 2020.

Fei Deng, Ingook Jang, and Sungjin Ahn. DreamerPro: Reconstruction-free model-based reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 4956–4975. PMLR, 2022.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on Robot Learning*, pp. 1–16. PMLR, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. PMLR, 2017.

Haotian Fu, Hongyao Tang, Jianye Hao, Chen Chen, Xidong Feng, Dong Li, and Wulong Liu. Towards effective context for meta-reinforcement learning: An approach based on contrastive learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7457–7465, 2021.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Jiaxian Guo, Mingming Gong, and Dacheng Tao. A relational intervention approach for unsupervised dynamics generalization in model-based reinforcement learning. In *International Conference on Learning Representations*, 2022.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *Advances in Neural Information Processing Systems*, 31, 2018.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in Neural Information Processing Systems*, 31, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with discrete world models. In *International Conference on Learning Representations*, 2021.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Philip A Knight. The sinkhorn–knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1):261–275, 2008.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pp. 5639–5650. PMLR, 2020.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 5757–5766. PMLR, 2020.

Hao Liu, Richard Socher, and Caiming Xiong. Taming maml: Efficient unbiased meta-reinforcement learning. In *International Conference on Machine Learning*, pp. 4061–4071. PMLR, 2019.

Cong Lu, Philip J Ball, Tim GJ Rudner, Jack Parker-Holder, Michael A Osborne, and Yee Whye Teh. Challenges and opportunities in offline reinforcement learning from visual observations. *arXiv preprint arXiv:2206.04779*, 2022.

Bogdan Mazoure, Ahmed M Ahmed, R Devon Hjelm, Andrey Kolobov, and Patrick MacAlpine. Cross-trajectory representation learning for zero-shot generalization in RL. In *International Conference on Learning Representations*, 2022.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Yunchen Pu, Zhe Gan, Ricardo Henao, Xin Yuan, Chunyuan Li, Andrew Stevens, and Lawrence Carin. Variational autoencoder for deep learning of images, labels and captions. *Advances in Neural Information Processing Systems*, 29, 2016.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, pp. 5331–5340. PMLR, 2019.

Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. ProMP: Proximal meta-policy search. In *International Conference on Learning Representations*, 2019.

Younggyo Seo, Kimin Lee, Ignasi Clavera Gilaberte, Thanard Kurutach, Jinwoo Shin, and Pieter Abbeel. Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

Junjie Wang, Qichao Zhang, and Dongbin Zhao. Dynamic-horizon model-based value estimation with latent imagination. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. doi: 10.1109/TNNLS.2022.3215788.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021a.

Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021b.

Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep RL via meta-learning. In *International Conference on Learning Representations*, 2020.

A APPENDIX

A.1 RELATED WORKS

Model-based RL with high-dimensional input. Model-based reinforcement learning from high-dimensional observation aims to learn latent representations and policies with latent dynamics models. World Model (Ha & Schmidhuber, 2018) maps the high-dimensional observations to latent space by VAE (Pu et al., 2016) and builds a recurrent latent dynamics model to evolve policy in latent imagination. PlaNet (Hafner et al., 2019) utilizes a Recurrent State-Space Model (RSSM) to learn the representation and latent dynamics jointly. The transition probability is modeled on the latent space instead of the original state space. Dreamer (Hafner et al., 2020) utilizes the RSSM to perform value gradient propagation through long-term imagination. DreamerV2 (Hafner et al., 2021) extends the Dreamer agent with discrete world model representations.

Self-supervised representation learning. Recent works in self-supervised learning show great potential to learn effective representations from high-dimensional data. One class of these methods learns effective features by comparing positive and negative examples (Oord et al., 2018; Chen et al., 2020; He et al., 2020). MoCo (He et al., 2020) further improves contrastive training by generating the representations from a momentum encoder instead of the trained network. However, these methods necessitate a greater number of negative samples, which demands large batch sizes or memory banks. To address this challenge, some works propose to learn the representations without discriminating between samples. BYOL (Grill et al., 2020) introduces a momentum encoder to the training network to provide target representations. SwAV (Caron et al., 2020) proposes that the embeddings be learned by matching them to a set of learned prototypes.

RL with auxiliary visual task. Recent works show that self-supervised representation learning techniques are able to improve the performance of visual reinforcement learning significantly. CURL (Laskin et al., 2020) exacts effective representations via contrastive learning and improves sample efficiency significantly over pixel-based methods. DrQ (Yarats et al., 2021b) proposes data-regularized Q-learning, which regularizes the Q-value over multiple image transformations for efficient policy learning. Proto-RL (Yarats et al., 2021a) conducts a prototypical self-supervised framework that ties representational learning with exploration through prototypes. CTRL (Mazoure et al., 2022) utilizes prototypes to cluster trajectory representations and encourages behavioral similarity between clusters nearby. DreamerPro (Deng et al., 2022) incorporates prototypes into Dreamer (Hafner et al., 2020) to benefit representation learning and enhance the robustness of reconstruction-free MBRL agents. Inspired by these prototypes-based RL methods, we construct a prototypical context learning framework that extracts temporally consistent contextual information to capture local dynamics efficiently.

Dynamics generalization in RL. Dynamics generalization aims to generalize the policy or the learned world model across a distribution of environments with varying transition dynamics. Meta-learning has been proposed to improve the generalization ability of RL agents across dynamics changes. Gradient-based meta-RL algorithms (Finn et al., 2017; Rothfuss et al., 2019; Liu et al., 2019; Gupta et al., 2018) learn an effective initialization and adapt policy parameters in new dynamics environments with few policy gradient updates. Context-based meta-RL algorithms (Rakelly et al., 2019; Zintgraf et al., 2020; Lee et al., 2020; Seo et al., 2020; Fu et al., 2021; Guo et al., 2022) learn contextual information to explicitly capture local dynamics and show great promise for generalization tasks in complex environments. However, the above methods are all investigated for low-dimensional input tasks but lack discussion for high-dimensional input tasks. We take one step further by developing an effective context-based latent dynamics model to solve the dynamics generalization problem with high-dimensional input.

A.2 ALGORITHM

The training pseudocode is given in Algorithm 1.

A.3 THE IMAGINATION PROCESS FOR POLICY LEARNING

As shown in Figure 5, with the prototypical context-aware dynamics model, we can imagine the latent trajectories by model rollout without any interaction with the environment. We can further

Algorithm 1 Prototypical Context-aware Dynamics (ProtoCAD)

Initialization: Number of random seed episodes N, collect interval C, batch size B, sequence length M, Number of prototypes K, temperature parameter T, imagination horizon H, episode length L, learning rate α
Collect dataset $\mathcal{D}$ with N episodes through the interaction with the environment `ENV` using random actions
Initialize world model parameters ϕ and θ, prototypes $\{c_k\}_{k=1}^{K}$, actor network parameters ψ, critic network parameters ξ
Initialize $\bar{\theta} = \theta$
while not converged **do**
  for $c = 1, \cdots, C$ **do**
    Sample B data sequences $\{(a_\tau, o_\tau, r_\tau)\}_{\tau=t-M}^{t} \sim \mathcal{D}$
    Perform data augmentation to obtain $\left\{\left(a_\tau, o_\tau^{(i)}, r_\tau\right)\right\}_{\tau=t-M}^{t}, i \in \{1, 2\}$
    Derive RSSM states, $h_\tau^{(i)} = f_\phi\left(h_{\tau-1}^{(i)}, z_{\tau-1}^{(i)}, a_{\tau-1}\right)$, $z_\tau^{(i)} \sim q_\phi\left(z_\tau^{(i)} \mid h_\tau^{(i)}, o_\tau^{(i)}\right)$
    Concatenate states $s_\tau^{(i)} = (h_\tau^{(i)}, z_\tau^{(i)})$
    Compute $u_\tau = f_\theta(s_\tau^{(1)})$ and $w_{t-M:t-1,k} = \text{softmax}\left(\frac{u_{t-M:t-1} \cdot c_k}{T}\right), k = 1, 2, \cdots, K$
    Compute $\bar{u}_\tau = f_{\bar{\theta}}(s_\tau^{(2)})$ and $(\bar{w}_{\tau,1}, \cdots, \bar{w}_{\tau,K}) = \text{Sinkhorn-Knopp}\left(\bar{u}_\tau, \{c_k\}_{k=1}^{K}\right)$
    Update ϕ, θ and $\{c_k\}_{k=1}^{K}$ using $\mathcal{J}_{\text{ProtoCAD}}$
    Update $\bar{\theta}$ by exponential moving average of θ
    Imagine trajectories $\{(s_\tau, a_\tau)\}_{\tau=t}^{t+H}$, $u_\tau = f_\theta(s_\tau)$, $e_\tau = \sum_{k=1}^{K} w_{\tau,k} \cdot c_k$
    Concatenate features $x_\tau = (s_\tau, u_\tau, e_\tau)$ and compute value estimates $V_\lambda(x_\tau)$
    Update actor network parameters $\psi \leftarrow \psi + \alpha \hat{\nabla}_\psi \mathcal{J}_{\text{Actor}}$
    Update critic network parameters $\xi \leftarrow \xi - \alpha \hat{\nabla}_\xi \mathcal{J}_{\text{Critic}}$
  end for
  $o_1 \leftarrow$ `ENV.reset()`
  for $t = 1, \cdots, L$ **do**
    Compute $h_t = f_\phi\left(h_{t-1}, z_{t-1}, a_{t-1}\right)$, $z_t \sim q_\phi\left(z_t \mid h_t, o_t\right)$ from history and $s_t = (h_t, z_t)$
    Compute $u_t = f_\theta(s_t)$ and $e_t = \sum_{k=1}^{K} \text{softmax}\left(\frac{u_t \cdot c_k}{T}\right) \cdot c_k$
    Get latent feature $x_t = (s_t, u_t, e_t)$ and get $a_t \sim \pi_\psi(a_t | x_t)$ with the actor
    Add exploration noise to action and execute it to get $r_t, o_{t+1} \leftarrow$ `ENV.step(`a_t`)`
  end for
  Add experience to dataset $\mathcal{D} \leftarrow \mathcal{D} \cup \{(a_t, o_t, r_t)\}_{t=1}^{L}$
end while

predict the rewards and values of future states and obtain the target values based on the context representation and latent states. The policy is optimized under the actor-critic framework.

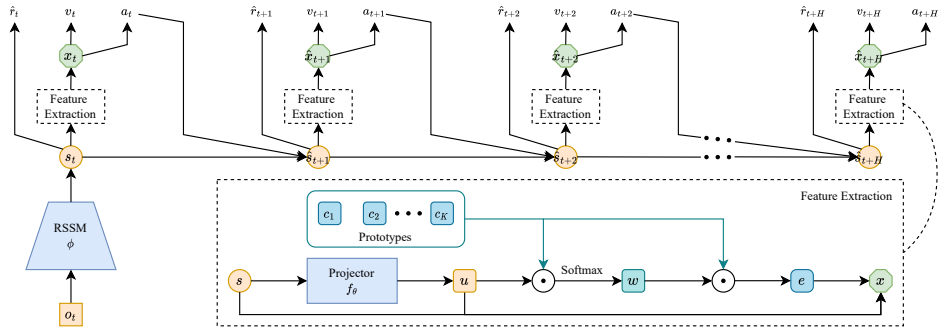


Figure 5: Latent trajectory imagination for policy learning.

A.4 Environmental settings

A.4.1 Discrete visual autonomous driving decision-making on CARLA

For the autonomous driving environment, we follow the main setting in DMVE (Wang et al., 2022). The CARLA (Dosovitskiy et al., 2017) simulator's traffic management tool is used to implement random traffic flow in the training and test scenarios. Screenshots of the CARLA driving environment are shown in Figure 6, where the ego vehicle is in black (the red line is the planned path) and other vehicles are in red. The observations are $64 \times 64 \times 3$ images corresponding to a local bird's eye view of the ego vehicle's range of 50m in front and 25m in back. An example of observation is given in Figure 7. Discrete action space is set as $\{$change lanes left, change lanes right, and stay in the current lane$\}$, and the reward function is designed to take safety and efficiency into account (Wang et al., 2022).

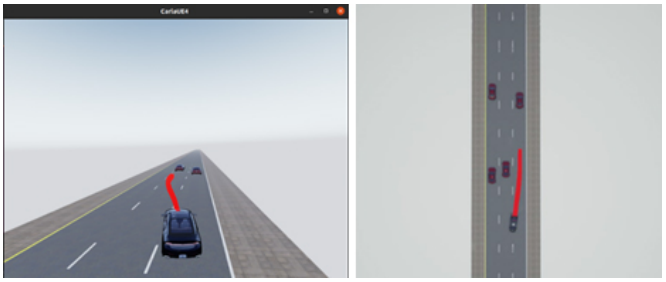


Figure 6: The CARLA driving environment (Wang et al., 2022).

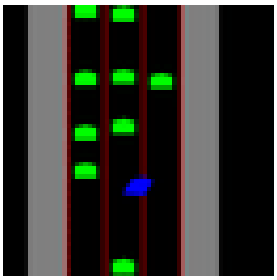


Figure 7: Observation example.

To construct the generalization environment, we generate traffic flows with different densities during training and testing. Specifically, during training, we generate 40–80 vehicles within a 1 km range in front of the ego vehicle and randomly generate 0–40 or 80–120 vehicles during testing. The desired speed of the ego vehicle is higher than the speed of the other vehicles.

A.4.2 Continuous visual control on DMC

The experimental setup of previous methods (Lee et al., 2020; Seo et al., 2020; Guo et al., 2022) on the dynamics generalization problem for state inputs provides us with a large number of references. In this paper, we follow many of the environment settings in RIA (Guo et al., 2022), with the difference that we modify the environment parameters in the DM-Control (DMC) (Tunyasuvunakool et al., 2020) benchmark with image-based observations, rather than the standard MuJoCo engine (Todorov et al., 2012) with state observations. Specifically, we develop our experimental environments on 6 different visual control tasks, including 1 DMC-Easy benchmark environment (i.e., Pendulum Swingup) and 5 DMC-Medium benchmark environments (i.e., Cheetah Run, Finger Spin, Hopper Hop, Quadruped Run, and Walker Run). We modify the mass m or damping d of the components in these environments and divide all parameter settings into a training set and a testing set. For training and testing, we sample the environment parameters at the beginning of each episode and keep them constant throughout that episode's interaction. The parameter settings during testing are not included in the training parameter set. As an example, the experiments in RIA vary the dynamics of HalfCheetah by modifying its rigid link mass and joint damping. Similarly, we achieve different dynamics in the Cheetah Run using the same training and testing parameters (the Cheetah environment in DMC versus the HalfCheetah in MuJoCo). In addition, we also supplement several environments (e.g., Finger and Walker) by referring to existing settings.

The specific environmental parameter settings are listed in Table 4.

A.5 Hyperparameters

For hyperparameters that are shared with DreamerPro (Deng et al., 2022), we use the default values suggested in the config file in the official implementation of DreamerPro. With the following two exceptions: we set the batch size as 16 as in Dreamer, and the number of prototypes K to be task-specified. The main hyperparameters are listed in Table 5 and Table 6.

Table 4: The environmental settings.

	Training Parameter List	Test Parameter List
Cheetah	$m \in \{0.75, 0.85, 1.00, 1.15, 1.25\}$ $d \in \{0.75, 0.85, 1.00, 1.15, 1.25\}$	$m \in \{0.2, 0.3, 0.4, 0.5, 1.5, 1.6, 1.7, 1.8\}$ $d \in \{0.2, 0.3, 0.4, 0.5, 1.5, 1.6, 1.7, 1.8\}$
Finger	$m \in \{0.5, 0.75, 1.0, 1.25, 1.5\}$	$m \in \{0.25, 0.375, 1.75, 2.0\}$
Hopper	$m \in \{0.5, 0.75, 1.0, 1.25, 1.5\}$	$m \in \{0.25, 0.375, 1.75, 2.0\}$
Pendulum	$m \in \{0.75, 0.8, 0.85, 0.9, 0.95, 1.0, 1.05, 1.1, 1.15, 1.2, 1.25\}$	$m \in \{0.2, 0.4, 0.5, 0.7, 1.3, 1.5, 1.6, 1.8\}$
Quadruped	$m \in \{0.85, 0.90, 0.95, 1.00\}$	$m \in \{0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60\}$
Walker	$m \in \{0.75, 0.85, 1.00, 1.15, 1.25\}$ $d \in \{0.75, 0.85, 1.00, 1.15, 1.25\}$	$m \in \{0.2, 0.3, 0.4, 0.5, 1.5, 1.6, 1.7, 1.8\}$ $d \in \{0.2, 0.3, 0.4, 0.5, 1.5, 1.6, 1.7, 1.8\}$

Table 5: Hyperparameters setting.

Hyperparameter	Meaning	Value
OP	Optimizer	Adam
N	Number of random seed episodes	2
B	Batch size	16
M	Sequence length	50
A	Action repeat	2
γ	Discount factor	0.99
α_w	Learning rate of the world model	3×10^{-4}
α_a	Learning rate of the actor model	8×10^{-5}
α_c	Learning rate of the critic model	8×10^{-5}
H	Imagination horizon	15
D	Prototype dimension	32
T	Softmax temperature	0.1
$SK - itr$	Sinkhorn-Knopp iterations	3
$SK - eps$	Sinkhorn-Knopp epsilon	0.05
η	Momentum update fraction	0.05

A.6 Additional results

A.6.1 Standard DMC comparison

Figure 8 shows the performance of our approach versus DreamerV2 (Hafner et al., 2021) on standard DMC tasks (without dynamics variation). For a fair comparison with DreamerV2, here all our model-related parameter settings are kept the same as its open-source code. The DreamerV2 results are from the original open-source repository. ProtoCAD outperforms DreamerV2 by a large margin. This also indicates that our approach works for different versions of the world model.

Table 6: Hyperparameter setting of the number of prototypes K.

Task	Value
Cheetah Run	100
Finger Spin	100
Hopper Hop	50
Pendulum Swingup	100
Quadruped Run	100
Walker Run	50

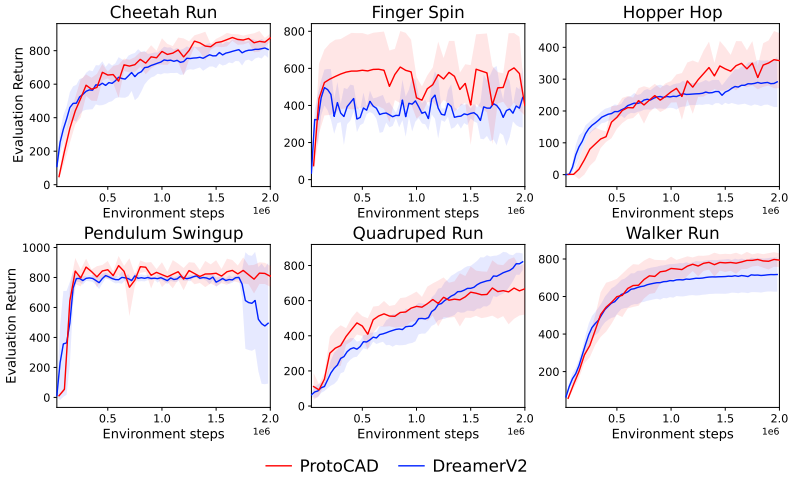


Figure 8: Performance comparison of ProtoCAD and DreamerV2 on standard DMC tasks (without dynamics variation). The mean and standard deviation of ProtoCAD are computed from 3 seeds.

A.6.2 TSNE VISUALIZATION

Our motivation is to extract the environment context information from the state trajectories produced by RSSM to assist in policy learning. Sinkhorn-Knopp can cluster trajectory data of batch and use prototypes to fit the different situations encountered in the learning process. The learned prototypes can characterize the context information. The results from TSNE (see Figure 9 and Figure 10) show that the learned context representation has some differentiated characterization results for different parameter settings. Also, the representation can be generalized to new parameter settings when testing under unseen environments. The learned representation is used as part of the input to the policy network and the value network so that the policy has some generalization ability in new environments as well.

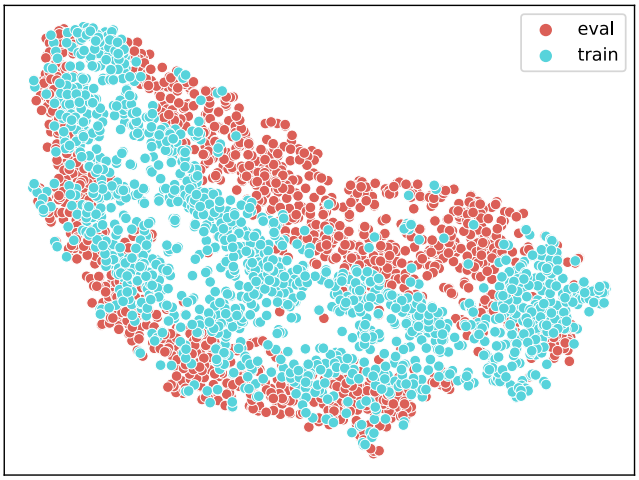
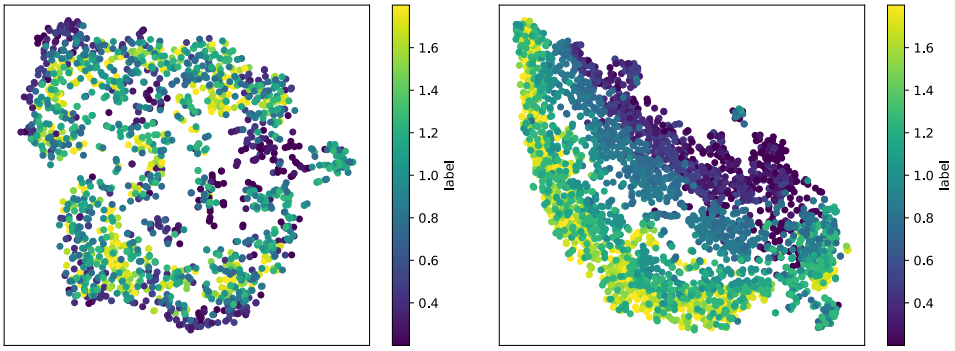


Figure 9: TSNE result of the learned feature for training and testing. After the training is completed, we extract features from the training and test data respectively, which is the concatenate of the projection output u and the aggregated prototypes based on the prediction probabilities w. We use TSNE to perform dimensionality reduction on this feature, and the visualization shows that the representation is in the vicinity of the training representation when tested in unseen environments, indicating that the context representation can be generalized.



(a) RSSM State (b) Learned Feature

Figure 10: TSNE results of RSSM state and learned feature for different parameter settings. Compared with the original RSSM state, the context representation has a significant clustering result for different parameter settings.