DreamerPro: Reconstruction-Free Model-Based Reinforcement Learning with Prototypical Representations

Fei Deng [1] Ingook Jang [2] Sungjin Ahn [3]

Abstract

Reconstruction-based Model-Based Reinforcement Learning (MBRL) agents, such as Dreamer, often fail to discard task-irrelevant visual distractions that are prevalent in natural scenes. In this paper, we propose a reconstruction-free MBRL agent, called DreamerPro, that can enhance robustness to distractions. Motivated by the recent success of prototypical representations, a non-contrastive self-supervised learning approach in computer vision, DreamerPro combines Dreamer with prototypes. In order for the prototypes to benefit temporal dynamics learning in MBRL, we propose to additionally learn the prototypes from the recurrent states of the world model, thereby distilling temporal structures from past observations and actions into the prototypes. Experiments on the DeepMind Control suite show that DreamerPro achieves better overall performance than state-of-the-art contrastive MBRL agents when there are complex background distractions, and maintains similar performance as Dreamer in standard tasks where contrastive MBRL agents can perform much worse.

1. Introduction

Model-Based Reinforcement Learning (MBRL, Sutton, 1991; Sutton & Barto, 2018) provides a solution to many problems in contemporary reinforcement learning. It improves sample efficiency by training a policy through simulations of a learned world model. Learning a world model also provides a way to efficiently represent experience data as general knowledge simulatable and reusable in arbitrary downstream tasks. In addition, it allows accurate and safe decisions via planning.

[1]Department of Computer Science, Rutgers University [2]ETRI [3]School of Computing, KAIST. Correspondence to: Fei Deng <fei.deng@rutgers.edu>, Sungjin Ahn <sungjin.ahn@kaist.ac.kr>.

Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022. Copyright 2022 by the author(s).

Among recent advances in image-based MBRL, Dreamer is particularly notable as the first MBRL agent outperforming popular model-free RL agents with better sample efficiency in both continuous and discrete control (Hafner et al., 2020; 2021). Unlike some previous MBRL methods (e.g., Kaiser et al., 2020), Dreamer learns a world model that can be rolled out in a compact latent representation space instead of the high-dimensional observation space. In addition, policy learning can be done efficiently via backpropagation through the differentiable dynamics model.

In image-based RL, the key problem is to learn a low-dimensional state representation and, in the model-based case, also its forward dynamics. While these can be learned by directly maximizing the rewards (Schrittwieser et al., 2020), due to reward sparsity, it is more practical to introduce auxiliary tasks that provide richer learning signal to facilitate representation learning (Sutton et al., 2011; Jaderberg et al., 2016). Dreamer achieves this by shaping the latent representation and dynamics in a way that reduces the reconstruction error of the observation sequences.

However, reconstruction-based representation learning has limitations. First, it wastes the representation capacity to learn even the visual signals that are irrelevant to the task such as noisy backgrounds (Zhang et al., 2021; Nguyen et al., 2021). Second, reconstruction can be computationally expensive for high-resolution observations, especially in models like Dreamer that need to reconstruct long-range videos. These limitations make it hard to scale Dreamer to natural scenes. Therefore, it is of particular interest to develop a reconstruction-free version of Dreamer for MBRL.

The currently dominant approaches for reconstruction-free MBRL optimize a contrastive loss over a batch of several consecutive frames (Okada & Taniguchi, 2021; Nguyen et al., 2021). Although they can greatly outperform Dreamer under complex background distractions, for the standard DeepMind Control (DMC) suite (Tassa et al., 2018), they showed quite inconsistent results by performing much worse than Dreamer on some tasks. Moreover, we find through visualization that in the presence of background distractions, the contrastively learned latent states can sometimes discard key task-relevant information. One possible reason is that the background distractions are visually different across the batch, leading to a degenerate solution that reduces the contrastive loss by distinguishing between the backgrounds.

In this paper, we seek to further enhance the robustness of reconstruction-free MBRL agents. Inspired by the prototypical representations that have shown more accurate and robust performance than contrastive methods in computer vision (Caron et al., 2020; 2021), we hypothesize that incorporating prototypes into MBRL so that they are jointly learned with the temporal dynamics, may also bring some benefits such as consistency or robustness. Hence, we propose DreamerPro, the first non-contrastive reconstruction-free MBRL agent that combines Dreamer with prototypes. Similar to SwAV (Caron et al., 2020), by encouraging uniform cluster assignment across the batch, we implicitly push apart the embeddings of different observations. Importantly, we set the number of prototypes as the product of batch size and sequence length, so that the embeddings within the same sequence (where the distractions tend to be visually similar) are also pushed apart. This helps avoid degenerate solutions and encourage extraction of task-relevant information. However, SwAV treats each observation independently, ignoring the temporal structures that are crucial in MBRL. To address this issue, DreamerPro additionally learns the prototypes from the recurrent states of the world model, thereby distilling temporal structures from the recurrent states into the prototypes.

We evaluate DreamerPro on the standard setting of DMC, and also on a natural background setting, where the background is replaced by natural videos irrelevant to the task. DreamerPro achieves better overall performance than state-of-the-art contrastive MBRL agents (Okada & Taniguchi, 2021; Nguyen et al., 2021) in natural background DMC, and maintains similar performance as Dreamer in standard DMC. Our visualization also shows that DreamerPro is better at retaining task-relevant information in both settings.

2. Preliminaries

In this section, we briefly introduce the world model and learning algorithms used in DreamerV2 (Hafner et al., 2021) which our model builds upon. To indicate the general Dreamer framework (Hafner et al., 2020; 2021), we omit its version number in the rest of the paper.

2.1. Reconstruction-Based World Model Learning

Dreamer learns a recurrent state-space model (RSSM, Hafner et al., 2019) to predict forward dynamics and rewards in partially observable environments. At each time step t, the agent receives an image observation o_t and a scalar reward r_t (obtained by previous actions $a_{<t}$). The agent then chooses an action a_t based on its policy. The RSSM models the observations, rewards, and transitions through a probabilistic generative process: $p(o_{1:T}, r_{1:T} \mid a_{1:T}) =$

$$\int \prod_{t=1}^{T} p(o_t \mid h_t, s_t)\, p(r_t \mid h_t, s_t)\, p(s_t \mid h_t)\, \mathrm{d}s_{1:T}\,, \tag{1}$$

where the latent variables $s_{1:T}$ are the agent states, and $h_t = \mathrm{GRU}(h_{t-1}, s_{t-1}, a_{t-1})$ is a deterministic encoding of $s_{<t}$ and $a_{<t}$ (Chung et al., 2014). To infer the agent states from past observations and actions, a variational encoder is introduced:

$$q(s_{1:T} \mid o_{1:T}, a_{1:T}) = \prod_{t=1}^{T} q(s_t \mid h_t, o_t)\,. \tag{2}$$

The training objective is to maximize the evidence lower bound (ELBO): $\mathcal{J}_{\text{Dreamer}} =$

$$\sum_{t=1}^{T} \mathbb{E}_q \Big[\underbrace{\log p(o_t \mid h_t, s_t)}_{\mathcal{J}_{\text{O}}^t} + \underbrace{\log p(r_t \mid h_t, s_t)}_{\mathcal{J}_{\text{R}}^t} - \underbrace{D_{\text{KL}}(q(s_t \mid h_t, o_t) \parallel p(s_t \mid h_t))}_{\mathcal{J}_{\text{KL}}^t} \Big]\,. \tag{3}$$

2.2. Policy Learning by Latent Imagination

Dreamer interleaves policy learning with world model learning. During policy learning, the world model is fixed, and an actor and a critic are trained cooperatively from the latent trajectories imagined by the world model. Specifically, the imagination starts at each non-terminal state $\hat{z}_t = [h_t, s_t]$ encountered during world model learning. Then, at each imagination step $t' \geq t$, an action is sampled from the actor's stochastic policy: $\hat{a}_{t'} \sim \pi(\hat{a}_{t'} \mid \hat{z}_{t'})$. The corresponding reward $\hat{r}_{t'+1}$ and next state $\hat{z}_{t'+1}$ are predicted by the learned world model. Given the imagined trajectories, the actor improves its policy by maximizing the λ-return (Schulman et al., 2016; Sutton & Barto, 2018) plus an entropy regularizer that encourages exploration, while the critic is trained to approximate the λ-return through a squared loss.

3. DreamerPro

To compute the Dreamer training objective, more specifically $\mathcal{J}_{\text{O}}^t$ in Equation 3, a decoder is required to reconstruct the image observation o_t from the state $z_t = [h_t, s_t]$. Because this reconstruction loss operates in pixel space where all pixels are weighted equally, Dreamer tends to allocate most of its capacity to modeling complex visual patterns that cover a large pixel area (e.g., backgrounds). This leads to poor task performance when those visual patterns are task irrelevant, as shown in previous work (Nguyen et al., 2021).

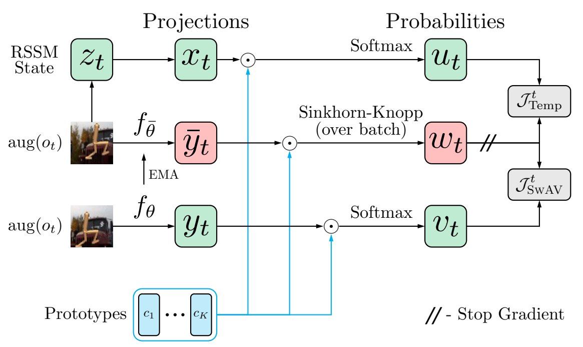

Figure 1. DreamerPro learns the world model through online clustering, eliminating the need for reconstruction. At each time step t, it first compares the observation to a set of trainable prototypes $\{c_1, \dots, c_K\}$ to obtain the target cluster assignment w_t. Then, it predicts this target from both the world model state z_t and another augmented view of the observation (each aug(o_t) denotes an independent application of data augmentation). The predictions are improved by optimizing the two objective terms, $\mathcal{J}^t_{\text{Temp}}$ and $\mathcal{J}^t_{\text{SwAV}}$, respectively, where the first term crucially distills temporal structures from z_t into the prototypes.

Fortunately, during policy learning, what we need is accurate reward and next state prediction, which are respectively encouraged by $\mathcal{J}^t_{\text{R}}$ and $\mathcal{J}^t_{\text{KL}}$. In other words, the decoder is not required for policy learning. The main purpose of having the decoder and the associated loss $\mathcal{J}^t_{\text{O}}$, as shown in Dreamer, is to learn meaningful representations that cannot be obtained by $\mathcal{J}^t_{\text{R}}$ and $\mathcal{J}^t_{\text{KL}}$ alone.

The above observations motivate us to improve robustness to visual distractions by replacing the reconstruction-based representation learning in Dreamer with reconstruction-free methods. To this end, we propose to combine Dreamer with the prototypical representations used in SwAV (Caron et al., 2020), a top-performing self-supervised representation learning method for static images. We name the resulting model DreamerPro, and provide the model description in the following.

DreamerPro uses the same policy learning algorithm as Dreamer, but learns the world model without reconstructing the observations. This is achieved by clustering the observation into a set of K trainable prototypes $\{c_1, \dots, c_K\}$, and then predicting the cluster assignment from the state as well as an augmented view of the observation. See Figure 1 for an illustration.

Concretely, given a sequence of observations $o_{1:T}$ sampled from the replay buffer, we obtain two augmented views $o^{(1)}_{1:T}, o^{(2)}_{1:T}$ by applying random shifts (Laskin et al., 2020a; Yarats et al., 2021b) with bilinear interpolation (Yarats et al., 2022). We ensure that the augmentation is consistent across time steps. Each view $i \in \{1, 2\}$ is fed to the RSSM to obtain the states $z^{(i)}_{1:T}$. To predict the cluster assignment from $z^{(i)}_t$, we first apply a linear projection followed by ℓ_2-normalization to obtain a vector $x^{(i)}_t$ of the same dimension as the prototypes, and then take a softmax over the dot products of $x^{(i)}_t$ and all the prototypes:

$$(u^{(i)}_{t,1}, \dots, u^{(i)}_{t,K}) = \text{softmax}\left(\frac{x^{(i)}_t \cdot c_1}{\tau}, \dots, \frac{x^{(i)}_t \cdot c_K}{\tau}\right). \tag{4}$$

Here, $u^{(i)}_{t,k}$ is the predicted probability that state $z^{(i)}_t$ maps to cluster k, τ is a temperature parameter, and the prototypes $\{c_1, \dots, c_K\}$ are also ℓ_2-normalized.

Analogously, to predict the cluster assignment from an augmented observation $o^{(i)}_t$, we feed it to a convolutional encoder (shared with the RSSM), apply a linear projection followed by ℓ_2-normalization, and obtain a vector $y^{(i)}_t$. We summarize this process as: $y^{(i)}_t = f_\theta(o^{(i)}_t)$, where θ collectively denotes the parameters of the convolutional encoder and the linear projection layer. The prediction probabilities are again given by a softmax:

$$(v^{(i)}_{t,1}, \dots, v^{(i)}_{t,K}) = \text{softmax}\left(\frac{y^{(i)}_t \cdot c_1}{\tau}, \dots, \frac{y^{(i)}_t \cdot c_K}{\tau}\right), \tag{5}$$

where $v^{(i)}_{t,k}$ is the predicted probability that observation $o^{(i)}_t$ maps to cluster k.

To obtain the targets for the above two predictions (i.e., Equations 4 and 5), we apply the Sinkhorn-Knopp algorithm (Cuturi, 2013) to the cluster assignment scores computed from the output of a momentum encoder $f_{\bar{\theta}}$ (He et al., 2020; Grill et al., 2020; Caron et al., 2021), whose parameters $\bar{\theta}$ are updated using the exponential moving average of θ: $\bar{\theta} \leftarrow (1-\eta)\bar{\theta} + \eta\theta$. For each observation $o^{(i)}_t$, the scores are given by the dot products $(\bar{y}^{(i)}_t \cdot c_1, \dots, \bar{y}^{(i)}_t \cdot c_K)$, where $\bar{y}^{(i)}_t = f_{\bar{\theta}}(o^{(i)}_t)$ is the momentum encoder output. The Sinkhorn-Knopp algorithm is applied to the two augmented batches $\{o^{(1)}_{1:T}\}, \{o^{(2)}_{1:T}\}$ separately to encourage uniform cluster assignment within each augmented batch and avoid trivial solutions. We specifically choose the number of prototypes $K = B \times T$, where B is the batch size and T is the sequence length, so that the observation embeddings are implicitly pushed apart from each other both within each sequence and across different sequences. The outcome of the Sinkhorn-Knopp algorithm is a set of cluster assignment targets $(w^{(i)}_{t,1}, \dots, w^{(i)}_{t,K})$ for each observation $o^{(i)}_t$.

Now that we have the cluster assignment predictions and targets, the representation learning objective is simply to maximize the prediction accuracies:

$$\mathcal{J}_{\mathrm{SwAV}}^t = \frac{1}{2}\sum_{k=1}^{K}\left(w_{t,k}^{(1)}\log v_{t,k}^{(2)} + w_{t,k}^{(2)}\log v_{t,k}^{(1)}\right) , \qquad (6)$$

$$\mathcal{J}_{\mathrm{Temp}}^t = \frac{1}{2}\sum_{k=1}^{K}\left(w_{t,k}^{(1)}\log u_{t,k}^{(1)} + w_{t,k}^{(2)}\log u_{t,k}^{(2)}\right) . \qquad (7)$$

Here, $\mathcal{J}_{\mathrm{SwAV}}^t$ improves prediction from an augmented view. This is the same loss as used in SwAV (Caron et al., 2020), and is shown to induce useful features for static images. However, it ignores the temporal structures that are crucial in MBRL. Hence, we add a second term, $\mathcal{J}_{\mathrm{Temp}}^t$, that improves prediction from the state of the same view. This has the effect of making the prototypes close to the states that summarize the past observations and actions, thereby distilling temporal structures into the prototypes. From another perspective, $\mathcal{J}_{\mathrm{Temp}}^t$ is similar to $\mathcal{J}_{\mathrm{O}}^t$ in the sense that we are now 'reconstructing' the cluster assignment of the observation instead of the observation itself. This frees the world model from modeling complex visual details, allowing more capacity to be devoted to task-relevant features.

The overall world model learning objective for DreamerPro can be obtained by replacing $\mathcal{J}_{\mathrm{O}}^t$ in Equation 3 with $\mathcal{J}_{\mathrm{SwAV}}^t + \mathcal{J}_{\mathrm{Temp}}^t$:

$$\mathcal{J}_{\mathrm{DreamerPro}} = \sum_{t=1}^{T}\mathbb{E}_q[\mathcal{J}_{\mathrm{SwAV}}^t + \mathcal{J}_{\mathrm{Temp}}^t + \mathcal{J}_{\mathrm{R}}^t - \mathcal{J}_{\mathrm{KL}}^t] , \qquad (8)$$

where $\mathcal{J}_{\mathrm{R}}^t$ and $\mathcal{J}_{\mathrm{KL}}^t$ are now averaged over the two augmented views.

4. Experiments

Environments. We evaluate our model and the baselines on six image-based continuous control tasks from the DeepMind Control (DMC) suite (Tassa et al., 2018). We choose the set of tasks based on those considered in PlaNet (Hafner et al., 2019). Specifically, we replace Cartpole Swingup and Walker Walk with their more challenging counterparts, Cartpole Swingup Sparse and Walker Run, and keep the remaining tasks. In addition to the standard setting, we also consider a natural background setting (Zhang et al., 2021; Nguyen et al., 2021), where the background is replaced by task-irrelevant natural videos randomly sampled from the 'driving car' class in the Kinetics 400 dataset (Kay et al., 2017). Following TPC (Nguyen et al., 2021), we use two separate sets of background videos for training and evaluation, containing 683 and 69 videos respectively. Hence, the natural background setting tests generalization to unseen distractions. We note that the recently released Distracting Control Suite (DCS, Stone et al., 2021) serves a similar purpose. However, the background distractions in DCS seem less challenging, as there are fewer videos and the ground plane is made visible for most tasks. In our preliminary experiments, our model and all the baselines achieved close to zero returns on Cartpole Swingup Sparse in the natural background setting. We therefore switch back to Cartpole Swingup in this setting.

Baselines. Our main baselines are Dreamer (Hafner et al., 2021), Dreaming (Okada & Taniguchi, 2021), and TPC (Nguyen et al., 2021), the state-of-the-art for reconstruction-based and reconstruction-free MBRL. In particular, TPC has shown better overall performance than CVRL (Ma et al., 2021), DBC (Zhang et al., 2021), and CURL (Laskin et al., 2020b) on the same datasets, and therefore is a strong baseline. The recently proposed PSE (Agarwal et al., 2021) has demonstrated impressive results on DCS. However, it is only shown to work in the model-free setting and requires a pretrained policy, while DreamerPro learns both the world model and the policy from scratch.

Implementation Details. We implement DreamerPro[1] and Dreaming based on a newer version of Dreamer[2], while the official implementation of TPC[3] is based on an older version. For fair comparison, we re-implement TPC based on the newer version. We adopt the default values for the Dreamer hyperparameters, except that we use continuous latents and `tanh_normal` as the distribution output by the actor. We find these changes improve Dreamer's performance in the standard DMC, and therefore use these values for all models in both the standard and the natural background settings. Following TPC, we increase the weight of the reward loss $\mathcal{J}_{\mathrm{R}}^t$ to 1000 for all models in the natural background setting to further encourage extraction of task-relevant information. While in the original TPC, this weight is chosen separately for each task from $\{100, 1000\}$, we find the weight of 1000 works consistently better in our re-implementation, which also obtains better results than reported in the original paper. We use the default batch size of 50 for Dreamer, Dreaming, and DreamerPro. The batch size for TPC is chosen to be 150, so that it has similar wall clock training time as DreamerPro.

Evaluation Protocol. For each task, we train each model for 1M environment steps (equivalent to 500K actor steps, as the action repeat is set to 2). The evaluation return is computed every 10K steps, and averaged over 10 episodes. In all figures and tables, the mean and standard deviation are computed from 3 independent runs.

[1] https://github.com/fdeng18/dreamer-pro

[2] https://github.com/danijar/dreamerv2/tree/e783832f01b2c845c195587158c4e129edabaebb

[3] https://github.com/VinAIResearch/TPC-tensorflow

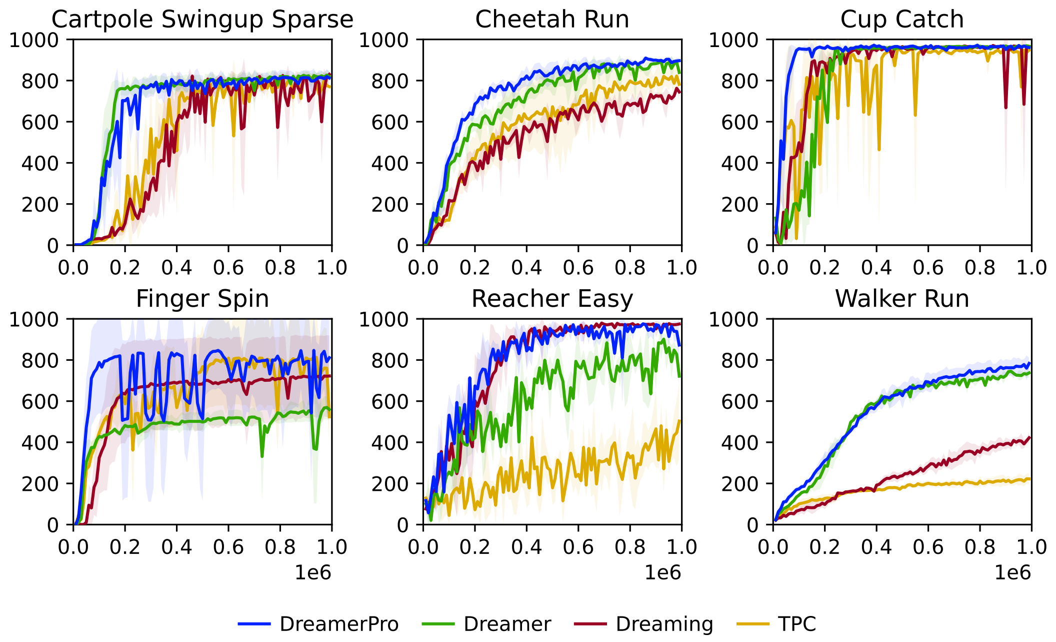

Figure 2. Performance curves in standard DMC. DreamerPro is the only model that is comparable or better than Dreamer on all tasks.

Table 1. Final performance in standard DMC.

Task	Dreamer	Dreaming	TPC	DreamerPro
Cartpole Swingup Sparse	$\mathbf{820 \pm 23}$	$\mathbf{830 \pm 12}$	770 ± 9	$\mathbf{813 \pm 32}$
Cheetah Run	840 ± 74	745 ± 18	782 ± 82	$\mathbf{897 \pm 8}$
Cup Catch	$\mathbf{967 \pm 3}$	$\mathbf{965 \pm 13}$	$\mathbf{948 \pm 7}$	$\mathbf{961 \pm 10}$
Finger Spin	559 ± 54	722 ± 197	524 ± 127	$\mathbf{811 \pm 232}$
Reacher Easy	721 ± 51	$\mathbf{975 \pm 2}$	503 ± 185	873 ± 127
Walker Run	$\mathbf{737 \pm 26}$	422 ± 25	222 ± 29	$\mathbf{784 \pm 28}$

4.1. Performance in Standard DMC

We show the performance curves in Figure 2 and the final performance in Table 1 for the standard setting. DreamerPro is the only model that achieves comparable or even better performance than Dreamer on all tasks. Notably, DreamerPro maintains similar performance as Dreamer on Walker Run where the other baselines failed. Moreover, DreamerPro demonstrates the best data efficiency on Cup Catch. We show through visualization in Section 4.3 that this is because DreamerPro can extract task-relevant information faster than the baselines. We also notice a large variance in DreamerPro's performance on Finger Spin. Further investigation reveals that DreamerPro learned close to optimal behavior (with average episode returns above 950) on two of the seeds, while converged to a suboptimal behavior (with average episode returns around 500) on the other seed. The low variance of Dreamer indicates that it hardly achieved close to optimal behavior. Our results suggest for the first time that prototypical representations can be beneficial to MBRL even in the absence of visual distractions.

4.2. Performance in Natural Background DMC

Figure 3 and Table 2 respectively show the performance curves and final evaluation returns obtained by all models in the natural background setting. Dreamer completely fails on all tasks, showing the inability of reconstruction-based representation learning to deal with complex visual distractions. In contrast, DreamerPro achieves better performance on 5 out of 6 tasks when compared to Dreaming and TPC individually. Considering the fact that contrastive methods tend to perform better in computer vision with larger batch sizes, we additionally train TPC with a batch size of 300 (denoted TPC-Batch-300). Among all the models, DreamerPro achieves the best performance on 4 out of 6 tasks, with large performance gains on Cartpole Swingup, Finger Spin, and Walker Run. These results indicate that the advantage of prototypical representations over contrastive learning in computer vision can indeed be transferred to MBRL for

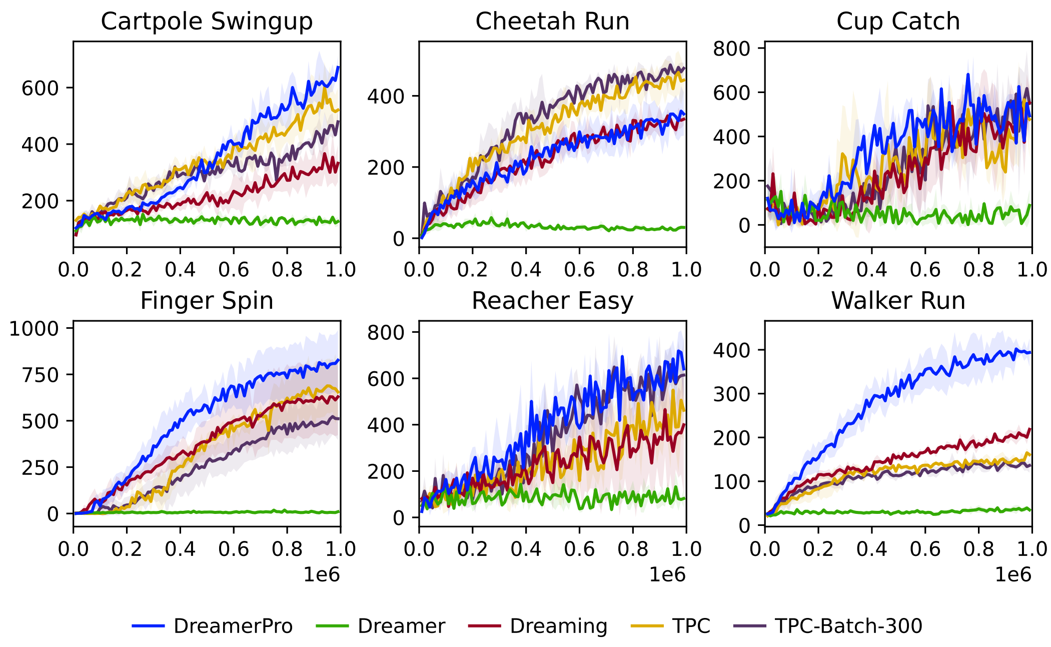

Figure 3. Performance curves in natural background DMC. DreamerPro significantly outperforms all baselines on Cartpole Swingup, Finger Spin, and Walker Run, while Dreamer completely fails on all tasks.

Table 2. Final performance in natural background DMC.

Task	Dreamer	Dreaming	TPC	TPC-Batch-300	DreamerPro
Cartpole Swingup	126 ± 16	332 ± 66	521 ± 80	479 ± 45	$\mathbf{671 \pm 42}$
Cheetah Run	30 ± 2	334 ± 17	$\mathbf{444 \pm 35}$	$\mathbf{477 \pm 16}$	349 ± 61
Cup Catch	88 ± 73	$\mathbf{553 \pm 60}$	477 ± 175	$\mathbf{550 \pm 69}$	493 ± 109
Finger Spin	10 ± 1	629 ± 207	655 ± 133	511 ± 115	$\mathbf{826 \pm 162}$
Reacher Easy	82 ± 39	400 ± 296	462 ± 130	$\mathbf{614 \pm 164}$	$\mathbf{641 \pm 123}$
Walker Run	35 ± 4	219 ± 9	161 ± 6	136 ± 17	$\mathbf{394 \pm 33}$

better robustness to visual distractions.

4.3. Visualization and Analysis

To better understand how the models work, we visualize the learned latent states through reconstruction from an auxiliary decoder, which is trained along with the agent learning process. For reconstruction-free methods, the world model does not receive gradients from this decoder.

Figure 4 (Left) shows the reconstructions for Cup Catch in the standard setting after 100K environment steps. We see that the reconstruction of the ball is only possible from the RSSM states learned by DreamerPro, explaining its better data efficiency. Figure 4 (Right) shows the reconstructions for Walker Run in the natural background setting after 1M environment steps. Notably, only DreamerPro is able to recover the posture of the Walker, demonstrating its superior ability to retain task-relevant information.

We additionally visualize the learned latent states through nearest neighbor queries in Figure 5 (more visualization can be found in Appendix J and K). Here, we first sample a batch of trajectories from the training replay buffer. Then, given a query image, we show the three images in the batch whose latent states are the closest to the query image. We use the same batch and same query images for all models.

It can be seen that the nearest neighbors for DreamerPro tend to have similar agent states as the query image, which is not the case for the other models. Meanwhile, DreamerPro can sometimes map similar backgrounds close to each other. This indicates that while DreamerPro prioritizes extracting task-relevant information, it may not fully discard the distractions. We note that this is less harmful than discarding task-relevant information as is done in Dreamer and TPC, because the policy network can also learn to ignore the distractions. Hence, we consider this a potential limitation to be addressed in future work.

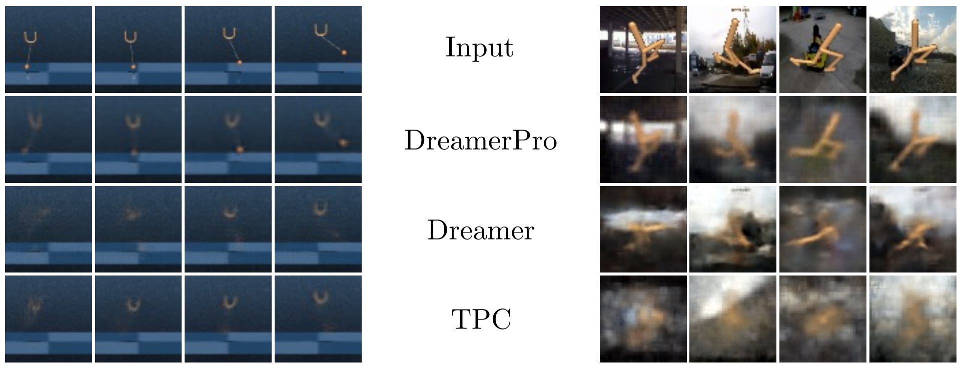

Figure 4. Visualization of learned latent states through reconstruction from an auxiliary decoder. The first row shows the input images, and the remaining rows show the reconstruction from the RSSM state z_t for each model.

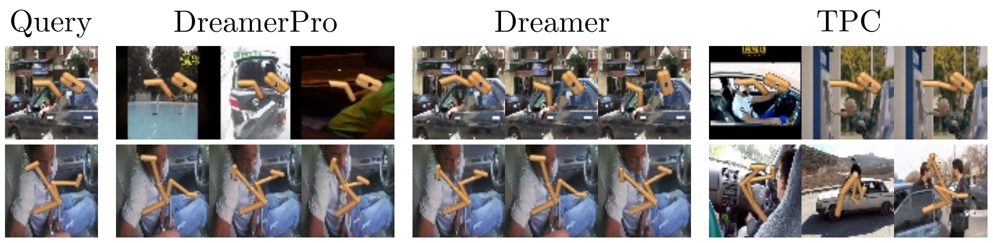

Figure 5. Visualization of learned latent states through nearest neighbor queries.

4.4. Ablation Study

We now show the individual effect of the two loss terms, $\mathcal{J}^t_{\text{SwAV}}$ and $\mathcal{J}^t_{\text{Temp}}$, in Figure 6 (full results are provided in Appendix B). Here, each of the ablated versions, DreamerPro-No-SwAV and DreamerPro-No-Temp, removes one of the loss terms. We did not investigate removing both terms, as its failure has been shown in Dreamer (Hafner et al., 2020). We train the ablated versions in natural background DMC, and observe that both terms are necessary for achieving good performance. In particular, naively combining SwAV with Dreamer (i.e., DreamerPro-No-Temp) leads to inferior performance, as it ignores the temporal structures. On the other hand, $\mathcal{J}^t_{\text{Temp}}$ alone is not sufficient to provide meaningful cluster assignment targets and learning signals for the convolutional encoder.

5. Related Work

Self-Supervised Representation Learning for Static Images. Recent work in self-supervised learning has shown effectiveness in learning representations from high-dimensional data. CPC (van den Oord et al., 2018) learns representations by maximizing the mutual information between the encoded representation and its future prediction using noise-contrastive estimation. SimCLR (Chen et al., 2020) shows that the contrastive data can be generated through random augmentations. MoCo (He et al., 2020), on the other hand, improves the contrastive training by generating the representations from a momentum encoder instead of the trained network. Despite the success in some tasks, one weakness of the contrastive approaches is that they require the model to compare a large number of samples, which demands large batch sizes or memory banks. To address this problem, some works propose to learn the image representations without discriminating between samples. Particularly, BYOL (Grill et al., 2020) introduces a momentum encoder to provide target representations for the training network. SwAV (Caron et al., 2020) proposes to learn the embeddings by matching them to a set of learnable clusters. DINO (Caron et al., 2021) replaces the clusters in SwAV with categorical heads and uses the centering and sharpening technique to prevent representational collapse. Unlike our model, these works treat each image independently and ignore the temporal structure of the environment, which is crucial in learning the forward dynamics and policy in MBRL.

Representation Learning for Model-Free Reinforcement Learning. It has been shown that adopting data augmenta-

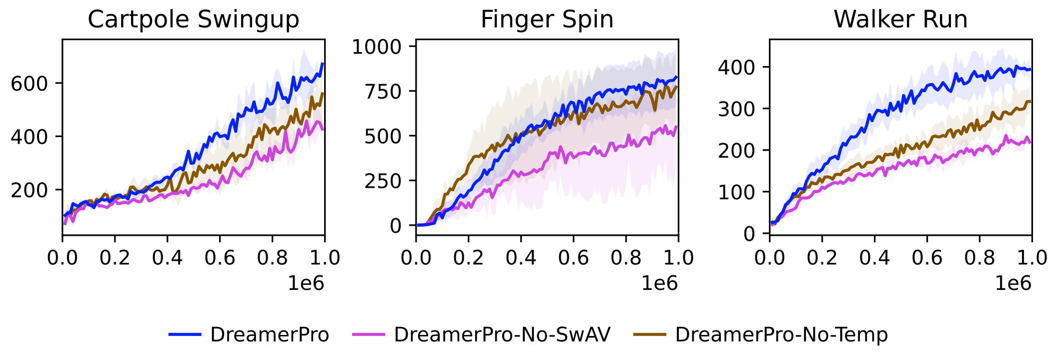

Figure 6. Ablation study. Both $\mathcal{J}^t_{\mathrm{SwAV}}$ and $\mathcal{J}^t_{\mathrm{Temp}}$ are necessary for achieving good performance.

tion techniques like random shifts in the observation space enables robust learning from pixel inputs in many model-free reinforcement learning algorithms (Laskin et al., 2020a; Yarats et al., 2021b; 2022). Recent works have also shown that self-supervised representation learning techniques can bring significant improvements to reinforcement learning methods. For example, CURL (Laskin et al., 2020b) performs contrastive learning along with off-policy RL algorithms and shows that it signficantly improves sample-efficiency and model performance over pixel-based methods. Other works aim to improve the representation learning quality by combining temporal prediction models in the representation learning process (Schwarzer et al., 2021a;b; Stooke et al., 2021; Yarats et al., 2021a; Guo et al., 2020; Gregor et al., 2019). However, the main purpose of the temporal prediction models in these works is to obtain the abstract representations of the observations, and they are not shown to support long-horizon imagination.

Model-Based Reinforcement Learning with Reconstruction. Model-based reinforcement learning from raw pixel data can learn the representation space by minimizing the observation reconstruction loss. World Models (Ha & Schmidhuber, 2018) learn the latent dynamics of the environment in a two-stage process to evolve their linear controllers in imagination. SOLAR (Zhang et al., 2019) models the dynamics as time-varying linear-Gaussian and solves robotic tasks via guided policy search. Dreamer (Hafner et al., 2020) jointly learns the RSSM and latent state space from observation reconstruction loss. DeepMDP (Gelada et al., 2019) also proposes a latent dynamics model-based method that uses bisimulation metrics and reconstruction loss in Atari. However, reconstruction-based methods are susceptible to distractions irrelevant to the task (Nguyen et al., 2021). Furthermore, in a few cases, the latent representation fails to reconstruct small task-relevant objects in the environment (Okada & Taniguchi, 2021).

Reinforcement Learning under Visual Distractions. A large body of work on robust representation learning focuses on contrastive objectives. For example, CVRL (Ma et al., 2021) proposes to learn representations from complex observations by maximizing the mutual information between an image and its corresponding embedding using contrastive objectives. However, the learning objective of CVRL encourages the representation model to learn as much information as possible, including task-irrelevant information. Dreaming (Okada & Taniguchi, 2021) and TPC (Nguyen et al., 2021) tackle this problem by incorporating a dynamics model and applying contrastive learning in the temporal dimension, which encourages the model to capture controllable and predictable information in the latent space. Bisimulation-based methods such as DBC (Zhang et al., 2021) and PSE (Agarwal et al., 2021) are another type of representation learning robust to visual distractions. Using the bisimulation metrics that quantify the behavioral similarity between states, these methods make the model robust to task-irrelevant information. However, DBC cannot generalize to unseen backgrounds (Nguyen et al., 2021), and PSE is only shown to work in the model-free setting and requires a pre-trained policy to compute the similarity metrics, while our model learns both the world model and the policy from scratch. Finally, TIA (Fu et al., 2021) is a reconstruction-based approach that adversarially trains a distractor model to separate the distractions from task-relevant information. Its main drawback is the sensitivity to hyperparameters—they need to be tuned for each task to balance the loss terms. In contrast, our model can use the same hyperparameter values across the DMC tasks.

6. Conclusion

In this work, we presented DreamerPro, the first non-contrastive reconstruction-free MBRL agent based on prototypical representations. In experiments on the standard and natural background DMC, DreamerPro showed better overall performance than state-of-the-art reconstruction-based and reconstruction-free MBRL agents, and demonstrated superior ability to extract task-relevant information. Future work can investigate ways to further discard distractions in the representation. It is also an interesting direction to improve reconstruction-free MBRL agents on Atari games, as our initial experiments (Appendix C) show that they still lag behind reconstruction-based MBRL agents. Lastly, prototypes have been demonstrated to help exploration in the model-free and distraction-free setting (Yarats et al., 2021a). Our model provides a starting point to extend such exploration to the model-based setting with distractions.

Acknowledgements

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant [22ZR1100, A Study of Hyper-Connected Thinking Internet Technology by autonomous connecting, controlling, and evolving ways] funded by the Korean government and Brain Pool Plus (BP+) Program (No. 2021H1D3A2A03103645) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT. The authors would like to thank Danijar Hafner, Masashi Okada, Tung Nguyen, and Xiang Fu for help with implementing Dreamer, Dreaming, TPC, and TIA respectively, Jindong Jiang and Ishani Ghose for editing the Related Work section, Chang Chen, Parikshit Bansal, and Jaesik Yoon for fruitful discussion, and anonymous reviewers for constructive comments.

References

Agarwal, R., Machado, M. C., Castro, P. S., and Bellemare, M. G. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9912–9924. Curran Associates, Inc., 2020.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9650–9660, October 2021.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Fu, X., Yang, G., Agrawal, P., and Jaakkola, T. Learning task informed abstractions. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3480–3491. PMLR, 18–24 Jul 2021.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. DeepMDP: Learning continuous latent space models for representation learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179. PMLR, 09–15 Jun 2019.

Gregor, K., Jimenez Rezende, D., Besse, F., Wu, Y., Merzic, H., and van den Oord, A. Shaping belief states with generative environment models for RL. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., and Valko, M. Bootstrap your own latent - A new approach to self-supervised learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21271–21284. Curran Associates, Inc., 2020.

Guo, Z. D., Pires, B. A., Piot, B., Grill, J.-B., Altché, F., Munos, R., and Azar, M. G. Bootstrap latent-predictive representations for multitask reinforcement learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3875–3886. PMLR, 13–18 Jul 2020.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference* *on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 09–15 Jun 2019.

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

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering Atari with discrete world models. In *International Conference on Learning Representations*, 2021.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., and Michalewski, H. Model based reinforcement learning for Atari. In *International Conference on Learning Representations*, 2020.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., et al. The Kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19884–19895. Curran Associates, Inc., 2020a.

Laskin, M., Srinivas, A., and Abbeel, P. CURL: Contrastive unsupervised representations for reinforcement learning. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 13–18 Jul 2020b.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. URLB: Unsupervised reinforcement learning benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

Ma, X., Chen, S., Hsu, D., and Lee, W. S. Contrastive variational reinforcement learning for complex observations. In Kober, J., Ramos, F., and Tomlin, C. (eds.), *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 959–972. PMLR, 16–18 Nov 2021.

Nguyen, T. D., Shu, R., Pham, T., Bui, H., and Ermon, S. Temporal predictive coding for model-based planning in latent space. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8130–8139. PMLR, 18–24 Jul 2021.

Okada, M. and Taniguchi, T. Dreaming: Model-based reinforcement learning by latent imagination without reconstruction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4209–4215, 2021.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, December 2020.

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

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2021a.

Schwarzer, M., Rajkumar, N., Noukhovitch, M., Anand, A., Charlin, L., Hjelm, R. D., Bachman, P., and Courville, A. C. Pretraining representations for data-efficient reinforcement learning. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 12686–12699. Curran Associates, Inc., 2021b.

Stone, A., Ramirez, O., Konolige, K., and Jonschkowski, R. The distracting control suite – A challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. Decoupling representation learning from reinforcement learning. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9870–9879. PMLR, 18–24 Jul 2021.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

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

Sutton, R. S., Modayil, J., Delp, M., Degris, T., Pilarski, P. M., White, A., and Precup, D. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 761–768, 2011.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. DeepMind Control Suite. *arXiv preprint arXiv:1801.00690*, 2018.

van den Oord, A., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11920–11931. PMLR, 18–24 Jul 2021a.

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

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Mastering visual continuous control: Improved data-augmented reinforcement learning. In *International Conference on Learning Representations*, 2022.

Zhang, A., McAllister, R. T., Calandra, R., Gal, Y., and Levine, S. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021.

Zhang, M., Vikram, S., Smith, L., Abbeel, P., Johnson, M., and Levine, S. SOLAR: Deep structured representations for model-based reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7444–7453. PMLR, 09–15 Jun 2019.

A. Hyperparameters

For hyperparameters that are shared with Dreamer, we use the default values suggested in the config file in the official implementation of Dreamer, with the following two exceptions. We set `rssm.discrete = False` and `actor.dist = tanh_normal`, as we find these changes improve performance over the default setting. The additional hyperparameters introduced in DreamerPro are listed in Table 3. We find it helpful to freeze the prototypes for the first 10K gradient updates. In the natural background setting, we add a squared loss that encourages the ℓ_2-norm of projections (before ℓ_2-normalization) to be close to 1. This helps stabilize the model.

Table 3. Additional hyperparameters in DreamerPro.

Hyperparameter	Value
Number of prototypes K	2500
Prototype dimension	32
Softmax temperature τ	0.1
Sinkhorn iterations	3
Sinkhorn epsilon	0.0125
Momentum update fraction η	0.05

B. Ablation Results

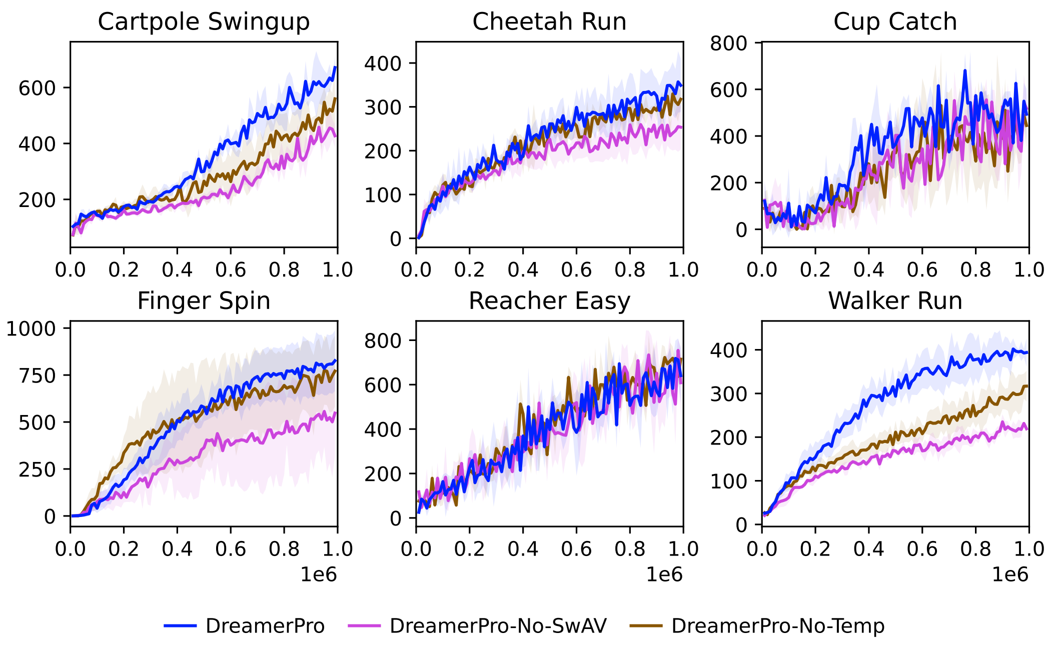

Figure 7. Full ablation results in natural background DMC.

C. Atari Results

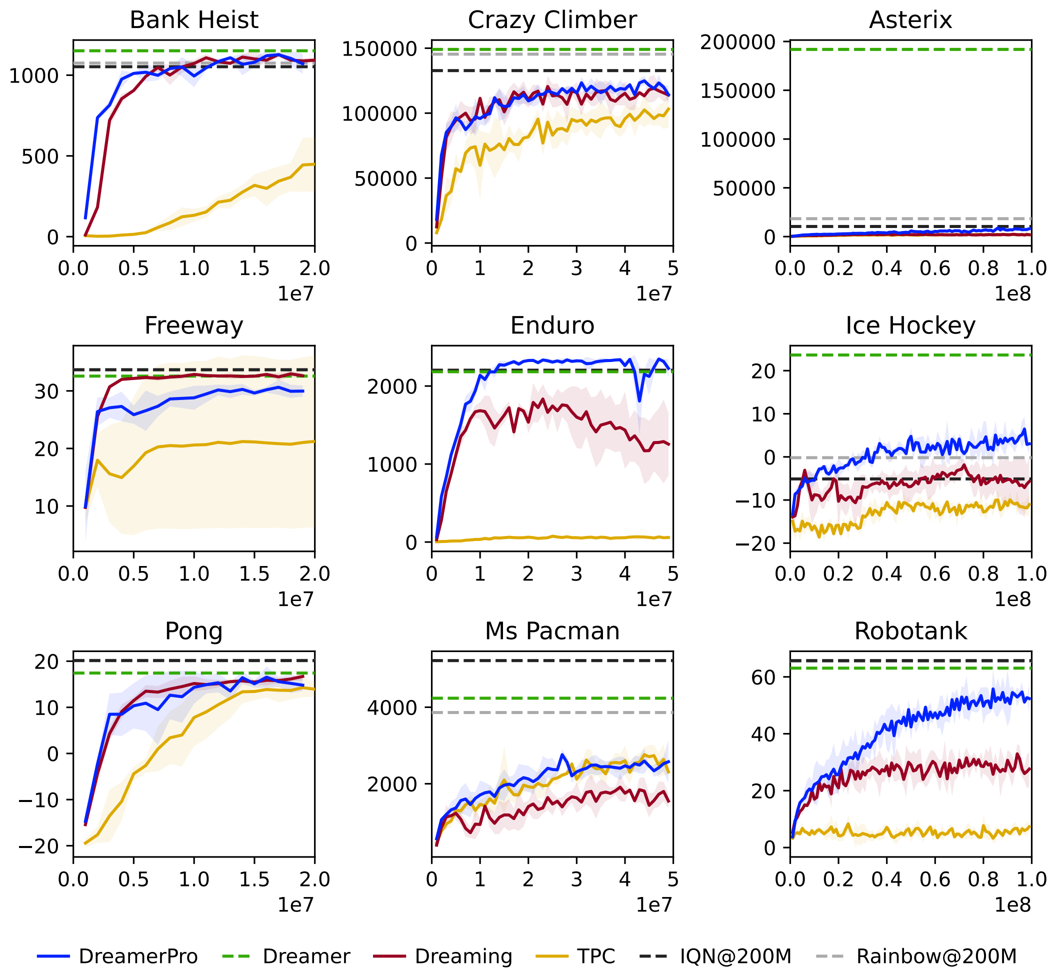

Figure 8. Performance curves on nine Atari games.

In this section, we compare DreamerPro with the baselines on a subset of nine Atari games. Our computational resources only allow us to run 3 seeds for each model on each game. Hence, we choose the games that have relatively low variance according to the performance curves in the Dreamer paper (Hafner et al., 2021). We divide the games into three categories based on Dreamer's convergence speed, i.e., 20M, 50M, and 100M environment steps, and pick three games from each category.

We note that both Dreaming and TPC have specific tricks that cannot be easily adapted to discrete latents. For fair comparison, we use continuous latents for all models. Following the author's suggestion, we set the dimension of stochastic state s_t to be 100, and keep other Dreamer hyperparameters as default. For Dreaming and TPC, we tuned the weight of reward loss from the set $\{1, 10, 100, 1000, 10000\}$, and used 10 for Dreaming and 1000 for TPC. We additionally tuned the number of future prediction steps in Dreaming from the set $\{3, 5\}$, and found 3 works better. For DreamerPro, we added a future prediction loss similar to what is used in SPR (Schwarzer et al., 2021a) and Dreaming. We also introduced an inverse dynamics loss (Schwarzer et al., 2021b), and found that it is helpful in Freeway. We freeze the prototypes for 30K gradient updates, and use the same hyperparameters as listed in Table 3.

Compared to Dreaming and TPC, DreamerPro performs significantly better on Enduro, Ice Hockey, and Robotank, while being comparable to the best model on the other games. However, DreamerPro still underperforms Dreamer on many games, indicating that Atari remains an unsolved challenge for reconstruction-free MBRL.

D. Additional Results in Standard DMC - Medium Benchmark

Figure 9 shows the performance curves of Dreamer and DreamerPro on the medium benchmark of standard DMC (Yarats et al., 2022). Here, for each task, we train each model for 3M environment steps. The evaluation return is computed every 10K steps, and averaged over 10 episodes. The mean and standard deviation are computed from 6 independent runs.

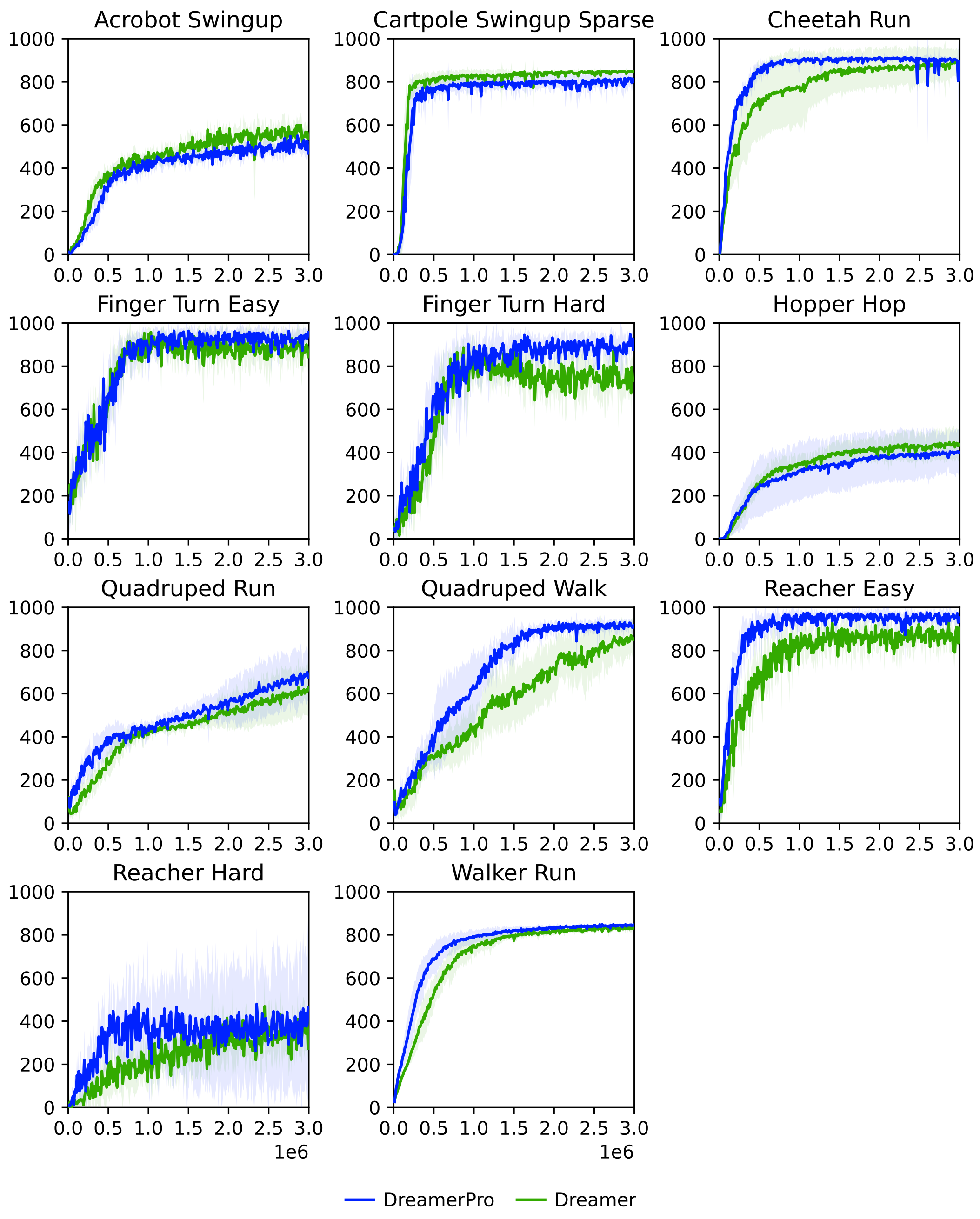

Figure 9. Performance curves in standard DMC - medium benchmark. DreamerPro is comparable or better than Dreamer on most tasks.

E. Additional Results in Standard DMC - Easy Benchmark

Figure 10 shows the performance curves of Dreamer and DreamerPro on the easy benchmark of standard DMC (Yarats et al., 2022). Here, for each task, we train each model for 1M environment steps. The evaluation return is computed every 10K steps, and averaged over 10 episodes. The mean and standard deviation are computed from 3 independent runs.

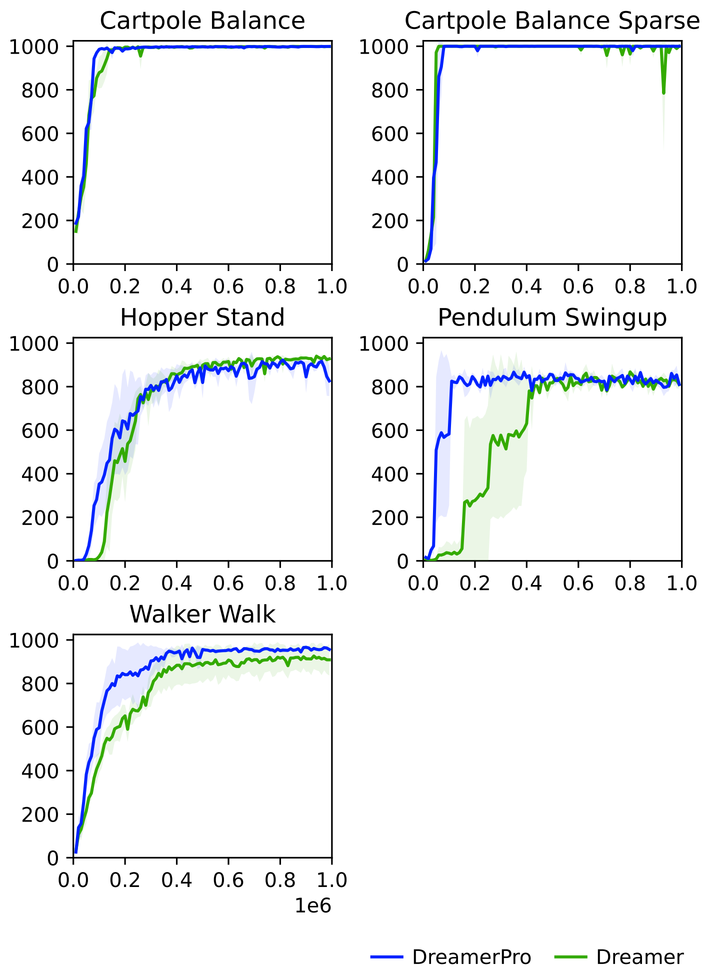

Figure 10. Performance curves in standard DMC - easy benchmark. DreamerPro is comparable or better than Dreamer on most tasks.

F. Model-Free Comparison

To see whether existing combinations of model-free RL with data augmentation and prototypes can achieve similar robustness to distractions, we report in Table 4 the performance of DrQ-v2 (Yarats et al., 2022) and Proto-RL (Yarats et al., 2021a) in the natural background DMC. We used the open-source implementation[4] in the Unsupervised RL Benchmark (Laskin et al., 2021), which incorporated the improvements of DrQ-v2 into Proto-RL. In addition to the default setting of Proto-RL that involves reward-free pretraining for 500K steps and then finetuning for 500K steps, we also experimented with training from scratch for 1M steps. In both settings, we combined intrinsic and extrinsic rewards, as this performed better than using extrinsic rewards alone. DrQ-v2 was trained for 1M steps. Our results in Table 4 show that DrQ-v2 and Proto-RL completely fail on many tasks when presented with distractions.

Table 4. Model-free comparison (natural background, 3 seeds).

Task	DreamerPro (ours)	DrQ-v2	Proto-RL	Proto-RL (no pretraining)
Cartpole Swingup	$\mathbf{671 \pm 42}$	314 ± 169	157 ± 17	179 ± 7
Cheetah Run	$\mathbf{349 \pm 61}$	69 ± 48	38 ± 8	28 ± 18
Cup Catch	493 ± 109	$\mathbf{560 \pm 87}$	150 ± 84	128 ± 162
Finger Spin	$\mathbf{826 \pm 162}$	76 ± 105	23 ± 21	52 ± 21
Reacher Easy	$\mathbf{641 \pm 123}$	268 ± 153	125 ± 40	97 ± 45
Walker Run	$\mathbf{394 \pm 33}$	108 ± 13	33 ± 6	48 ± 6

G. Effect of the Number of Prototypes

Our preliminary experiments suggest that performance generally improves with more prototypes (see Table 5 below). Hence, we use the maximum number of prototypes for a given batch. We expect that further increasing the number could be beneficial. However, that would require maintaining a cache of previous batches, which we did not consider for simplicity.

Table 5. Effect of #prototypes in Walker Run (natural background, 1M steps).

#Prototypes	50 (1 seed)	500 (1 seed)	1250 (1 seed)	2500 (3 seeds)
Episode Return	267	313	425	394 ± 33

H. Training Speed Comparison

While the speed of Dreamer and DreamerPro at test time is similar, the training speed of Dreamer can be significantly lower as the image resolution increases. In Table 6 below, we record during training the number of frames processed per second (FPS) by Dreamer and DreamerPro on NVIDIA Quadro RTX 8000 GPUs. We made reasonable changes to the encoder and decoder, adding one convolutional layer each time the image size is doubled. Other hyperparameters were kept fixed. We used mixed precision in all settings except when training Dreamer on 256×256 observations (due to numerical instability). Our results suggest that when observations are in high resolution (which is often desired in real world applications), the computational overhead caused by augmentation and prototypes is far less than reconstruction.

Table 6. Training FPS comparison on standard Walker Run.

Image Size	64×64	128×128	256×256
Dreamer	48	26	3
DreamerPro	44	25	9

[4]`https://github.com/rll-research/url_benchmark`

I. Additional Visualization via Reconstruction

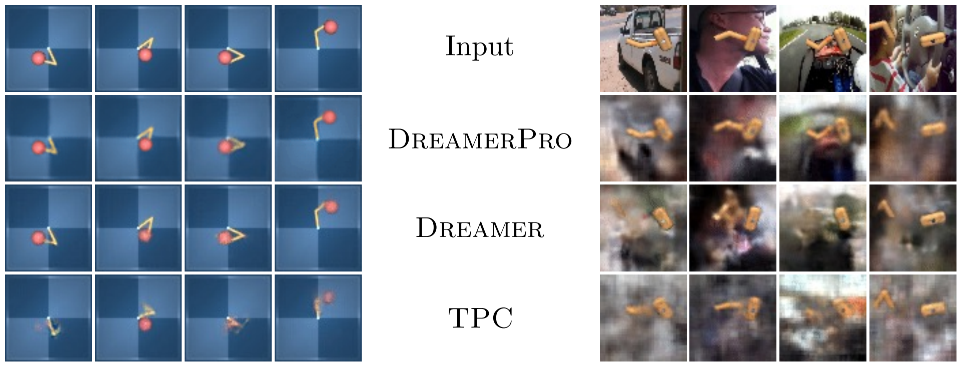

Figure 11. Visualization of learned latent states through reconstruction from an auxiliary decoder.

J. Additional Results of Nearest Neighbor Queries

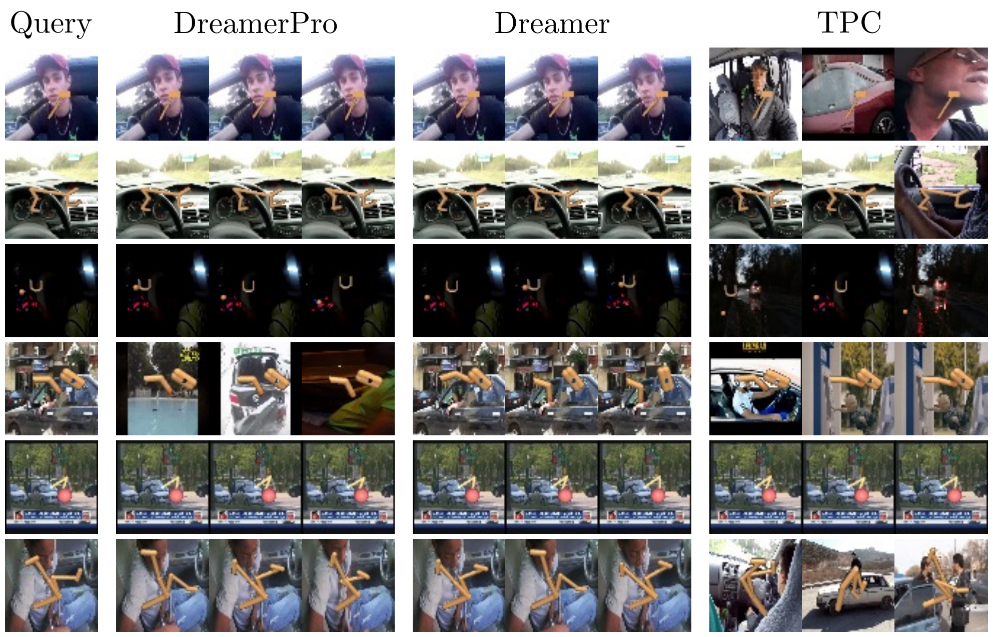

Figure 12. Visualization of learned latent states through nearest neighbor queries.

K. Prototype Visualizations

We visualize the first 5 prototypes learned for each task in natural background DMC using nearest neighbor queries, shown in each row of Figures 13 - 18. To do so, we sample a batch of trajectories from the training replay buffer, and obtain the projection x_t from latent state z_t for each image. Then, given a prototype c_k as query, we show the ten images in the batch whose projections are the closest to the prototype. Consistent with our findings in Section 4.3, some prototypes are able to capture specific agent states under different backgrounds, while others also map similar backgrounds close together.

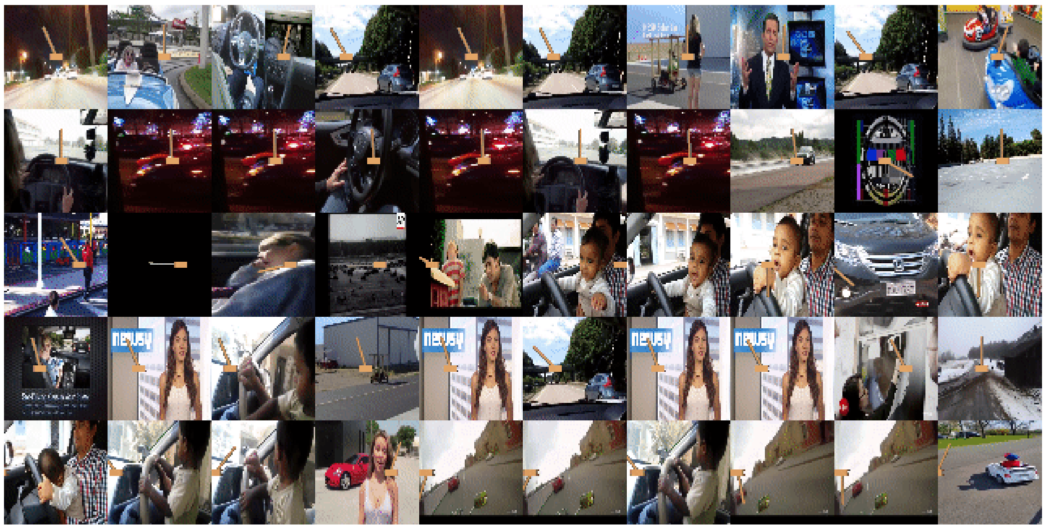

Figure 13. Prototype visualization for Cartpole Swingup.

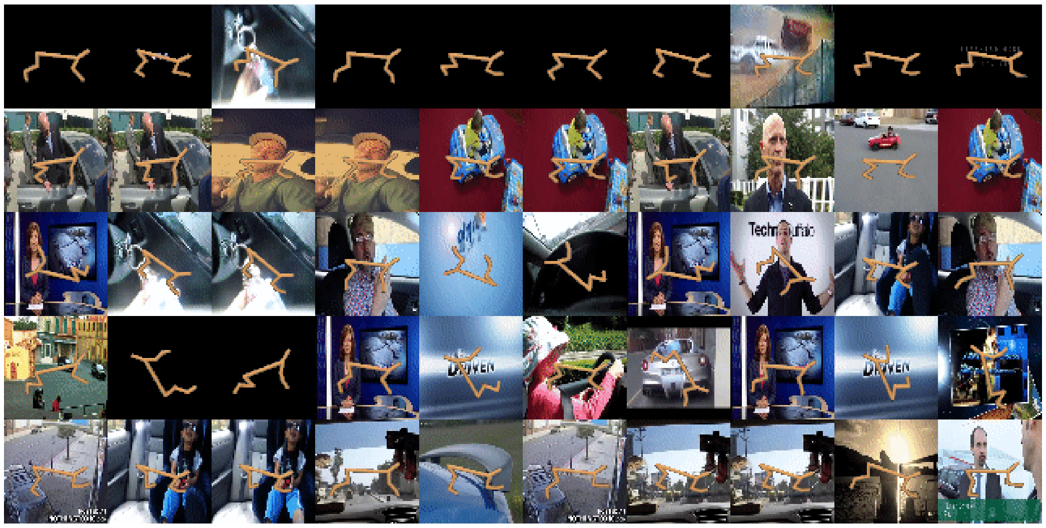

Figure 14. Prototype visualization for Cheetah Run.

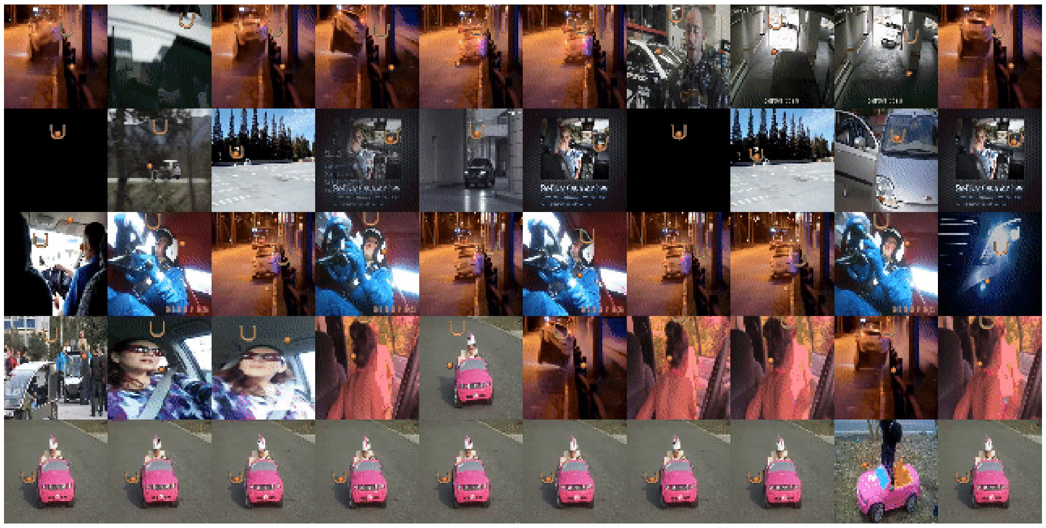

Figure 15. Prototype visualization for Cup Catch.

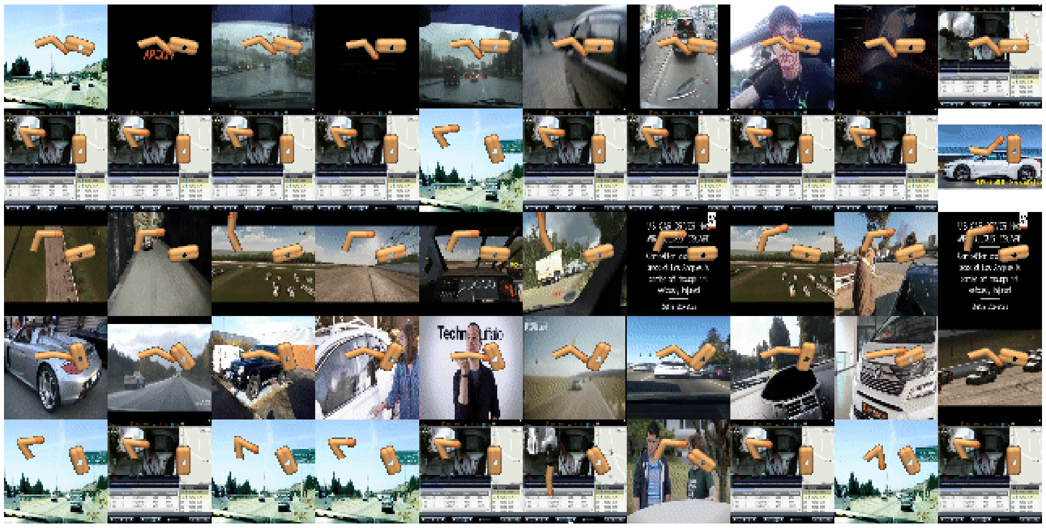

Figure 16. Prototype visualization for Finger Spin.

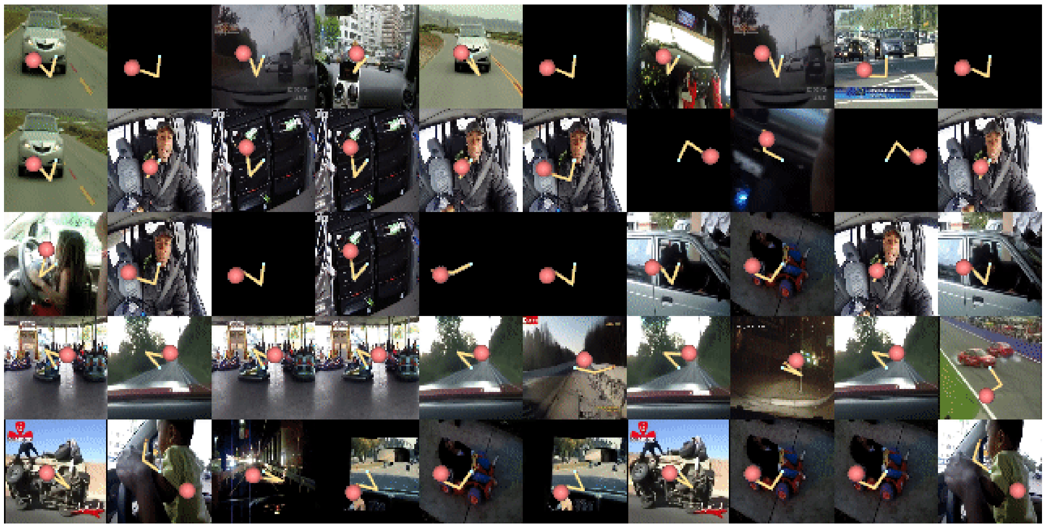

Figure 17. Prototype visualization for Reacher Easy.

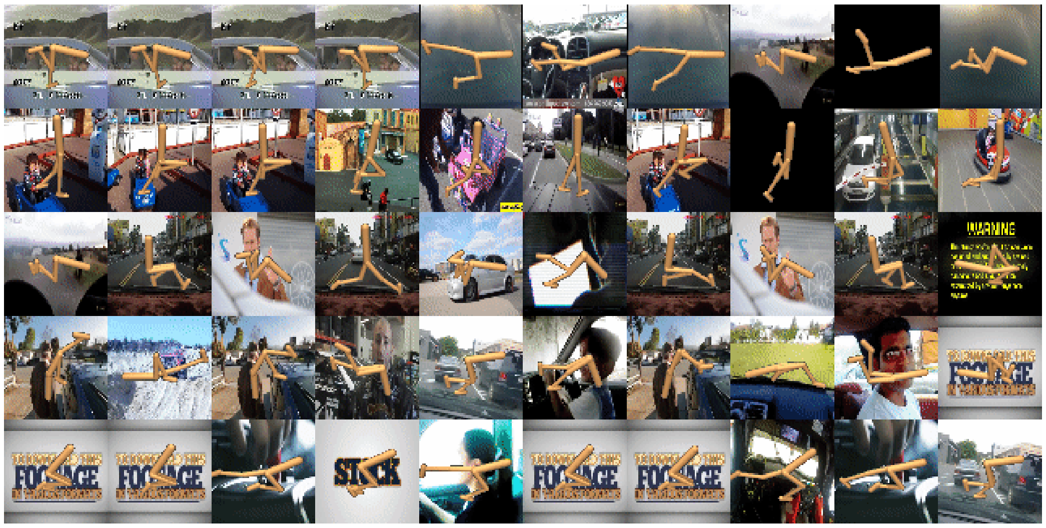

Figure 18. Prototype visualization for Walker Run.