Understanding Effectiveness of Learning Behavioral Metrics in Deep Reinforcement Learning

Anonymous authors
Paper under double-blind review

Keywords: behavioral metrics, bisimulation metrics, representation learning, evaluation

Summary

A key approach to state abstraction is approximating *behavioral metrics* (notably, bisimulation metrics) in the observation space, and embed these learned distances in the representation space. While promising for robustness to task-irrelevant noise shown in prior work, accurately estimating these metrics remains challenging, requiring various design choices that create gaps between theory and practice. Prior evaluations focus mainly on final returns, leaving the quality of learned metrics and the source of performance gains unclear. To systematically assess how metric learning works in deep RL, we evaluate five recent approaches. We unify them under isometric embedding, identify key design choices, and benchmark them with baselines across 20 state-based and 14 pixel-based tasks, spanning *250+ configurations* with diverse noise settings. Beyond final returns, we introduce the denoising factor to quantify the encoder's ability to filter distractions. To further isolate the effect of metric learning, we propose an isolated metric estimation setting, where the encoder is influenced solely by the metric loss. Our results show that metric learning improves return and denoising only marginally, as its benefits fade when key design choices, such as *layer normalization* and *self-prediction loss*, are incorporated into the baseline. We also find that commonly used benchmarks (e.g., grayscale videos, varying state-based Gaussian noise dimensions) add little difficulty, while Gaussian noise with random projection and pixel-based Gaussian noise remain challenging even for the best methods. Finally, we release an open-source, modular codebase to improve reproducibility and support future research on metric learning in deep RL.

Contribution(s)

1. We analyze five metric learning approaches under the isometric embedding framework to identify key design choices.
   Context: Metric learning methods often diverge significantly between theory and implementation.
2. We introduce the denoising factor to quantify an encoder's ability to filter distractions.
   Context: Metric learning is often motivated by denoising ability but is rarely evaluated directly, with prior work relying mainly on qualitative analysis (Zhang et al., 2020).
3. We benchmark five metric learning approaches across diverse distracting domains and find that common benchmarks add little difficulty to clean tasks, while certain noise settings remain challenging even for the best methods.
   Context: Prior work primarily uses IID Gaussian noise with varied dimensions (Ni et al., 2024) and grayscale video backgrounds (Zhang et al., 2020).
4. Through ablation studies, we identify layer normalization and self-prediction loss as key design choices across all methods.
   Context: Prior work in metric learning does not isolate the effect of self-prediction loss and only shows the benefits of normalization in specific methods (Zang et al., 2022).
5. We show that the benefits of metric learning diminish in both return and denoising factor when key design choices are incorporated into the baseline.
   Context: Prior work does not report this limitation of metric learning.

Understanding Effectiveness of Learning Behavioral Metrics in Deep Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

A key approach to state abstraction is approximating *behavioral metrics* (notably, bisimulation metrics) in the observation space, and embed these learned distances in the representation space. While promising for robustness to task-irrelevant noise shown in prior work, accurately estimating these metrics remains challenging, requiring various design choices that create gaps between theory and practice. Prior evaluations focus mainly on final returns, leaving the quality of learned metrics and the source of performance gains unclear. To systematically assess how metric learning works in deep RL, we evaluate five recent approaches. We unify them under isometric embedding, identify key design choices, and benchmark them with baselines across 20 state-based and 14 pixel-based tasks, spanning *250+ configurations* with diverse noise settings. Beyond final returns, we introduce the denoising factor to quantify the encoder's ability to filter distractions. To further isolate the effect of metric learning, we propose an isolated metric estimation setting, where the encoder is influenced solely by the metric loss.

Our results show that metric learning improves return and denoising only marginally, as its benefits fade when key design choices, such as *layer normalization* and *self-prediction loss*, are incorporated into the baseline. We also find that commonly used benchmarks (e.g., grayscale videos, varying state-based Gaussian noise dimensions) add little difficulty, while Gaussian noise with random projection and pixel-based Gaussian noise remain challenging even for the best methods. Finally, we release an open-source, modular codebase to improve reproducibility and support future research on metric learning in deep RL.[1]

1 Introduction

Real-world environments often present high-dimensional, noisy observations, posing challenges for RL. For instance, in image-based settings, task-irrelevant variations in background, lighting, and viewpoint introduce distractions. Yet, despite this observational complexity, system dynamics are typically governed by *a compact, task-relevant state*. State abstraction (Li et al., 2006; Konidaris, 2019) provides a framework for extracting such *latent representations* from raw observations, filtering out irrelevant information while preserving task-critical structure. A key principle of state abstraction is that behaviorally similar states should have similar representations. Traditionally, this is enforced through state aggregation (Singh et al., 1994; Givan et al., 2003), grouping states into discrete abstract classes based on equivalence relations. However, state aggregation lacks a measure of *how different* states are across classes and struggles with continuous representations, requiring infinitely many discrete classes.

To address this, *bisimulation metrics* (Ferns et al., 2004; 2011) and their scalable variants (Castro, 2020; Zhang et al., 2020) have been proposed to define meaningful distances between observations. These fall under the broader class of **behavioral metrics** (Castro et al., 2023), which quantify state

[1]The artifact is available at https://anonymous.4open.science/r/understanding_metric-3C44

similarity based on differences in immediate rewards and transition probabilities. By learning a metric alongside deep RL, prior work (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021; Chen & Pan, 2022; Zang et al., 2022) has shown progress in tackling high-dimensional, noisy tasks.

Nevertheless, the role of behavioral metric learning in deep RL (**metric learning** for short) remains unclear due to the lack of systematic evaluation. First, its effectiveness relies on accurately estimating these metrics, which is challenging in complex tasks. However, prior work primarily measures performance through *returns*, without directly assessing the quality of the learned metrics. Second, metric learning is often combined with *multiple losses* (e.g., self-prediction (Zhang et al., 2020), inverse dynamics (Kemertas & Aumentado-Armstrong, 2021)), as well as *architectural choices* (e.g., normalization, ensembles (Zang et al., 2022)), making it difficult to isolate the contribution of metric learning to performance gains. Third, most studies evaluate only *OOD generalization* in environments with *grayscale natural videos* as distractions (Zhang et al., 2020), conflating robustness with generalization. Lastly, prior evaluations (Tomar et al., 2021; Li et al., 2022) report inconsistent results for the same algorithms, raising concerns about reproducibility.

In this paper, we provide a understanding of **how metric learning works in deep RL** through a systematic evaluation of five recent approaches alongside two baselines. First, we unify these metric learning objectives under a common framework using the notion of isometric embedding, identifying key design choices for our investigation. Next, to ensure a rigorous and comprehensive evaluation, we introduce diverse distraction benchmarks by varying difficulty levels, from Gaussian noise to colored natural videos, across both state-based and pixel-based domains, tested under both ID and OOD generalization. Then, we quantify the **denoising** capability – the encoder's ability to filter out distractions. We introduce the *denoising factor*, which numerically measures how well the encoder distinguishes similar from dissimilar observations.[2] Finally, we propose an *isolated* metric estimation setting to assess metric learning's contribution to denoising, independent of other losses.

Contributions. Our main contributions are as follows:

1. **Conceptual**: We analyze five recent metric learning approaches using the isometric embedding framework to identify key design choices. In addition, we propose *denoising factor* to quantify an agent's denoising capability in distracting tasks.
2. **Comprehensive benchmarking**: We benchmark these five metric learning methods and baselines on diverse distracting variants of the DeepMind Control (DMC) suite (Tassa et al., 2018). In state-based domains, across 20 DMC tasks with 10 IID Gaussian noise settings, SimSR (Zang et al., 2022), originally evaluated in pixel-based domains, significantly outperforms other methods in both return and denoising factor. In pixel-based domains, across 14 DMC tasks with 6 image background settings, RAP (Chen & Pan, 2022) performs generally best. SAC (Haarnoja et al., 2018) and DeepMDP (Gelada et al., 2019) remain competitive baselines but are often overlooked.
3. **Reevaluating benchmark difficulty**: Surprisingly, common distracting benchmarks – varying Gaussian noise dimensions in state-based domains and grayscale videos in pixel-based domains – add little difficulty. However, Gaussian noise with random projection in state-based domains and Gaussian noise in pixel-based domains remain challenging, even for the best methods.
4. **Identifying key design choices**: We find **self-prediction loss** is crucial to SimSR's success. Notably, **(layer) normalization**, used in SimSR, consistently improves return and denoising across all metric learning methods and baselines.
5. **Marginal impact of metric learning (a bitter lesson)**: The benefits of metric learning diminish when key design choices are incorporated into the baseline.
6. **Open-source codebase**: We open-source a modular and efficient codebase to improve reproducibility in the RL community.

[2] Prior work (Zhang et al., 2020) qualitatively analyzes denoising by visualizing representations with t-SNE.

2 Background

2.1 Problem Formulation

We consider a setting where observations contain distractions and focus on a special class of Markov decision processes – exogenous block MDPs (EX-BMDPs) (Efroni et al., 2021; Islam et al., 2022). Before introducing EX-BMDPs, we first define block MDPs as a prerequisite.

Block MDPs (Du et al., 2019). A block MDP (BMDP) is a tuple $\langle \mathcal{X}, \mathcal{Z}, \mathcal{A}, q, p, R, \gamma \rangle$, where $\mathcal{X}$ is the observation space, $\mathcal{Z}$ is the latent state space, $\mathcal{A}$ is the action space, $p : \mathcal{Z} \times \mathcal{A} \to \Delta(\mathcal{Z})$ is latent transition function, $R : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ is (latent) reward function, and $\gamma \in [0, 1)$ is the discount factor. The emission function $q : \mathcal{Z} \to \Delta(\mathcal{X})$ generates observation $x \sim q(\cdot \mid z)$ from latent state z. Crucially, BMDP assumes the *block structure*: $\forall\, z_1, z_2 \in \mathcal{Z},\ z_1 \neq z_2 \implies \text{supp}(q(\cdot \mid z_1)) \cap \text{supp}(q(\cdot \mid z_2)) = \emptyset$. This ensures that each observation uniquely determines its latent state, enabling the existence of the *oracle encoder* $q^{-1} : \mathcal{X} \to \mathcal{Z}$ such that $q^{-1}(x) = z$ whenever $x \sim q(\cdot \mid z)$. The goal of RL in BMDP is to find a policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ that maximizes the rewards: $\max_\pi \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(q^{-1}(x_t), a_t)\right]$. The policy only receives the observation x without access to the latent state z, the latent space $\mathcal{Z}$, or the oracle encoder q^{-1}. While the class of BMDPs is equivalent to the class of MDPs (Du et al., 2019), they capture the underlying state from a high-dimensional observation. However, BMDPs do not differentiate between task-relevant (endogenous) state and task-irrelevant (exogenous) noise in the latent space.

Exogenous BMDPs (Efroni et al., 2021). An EX-BMDP extends BMDP by factorizing a latent state into $z = (s, \xi)$, where $s \in \mathcal{S}$ is the *task-relevant state* and $\xi \in \Xi$ is the *task-irrelevant noise*, representing distraction. The latent state transition $p(s', \xi' \mid s, \xi, a)$ factorizes as $p(s' \mid s, a)p(\xi' \mid \xi)$, where the noise ξ evolves independently and does not affect the reward function. To simplify notation, we denote the reward function as as $R(s, a)$. EX-BMDPs guarantee the existence of a denoising map $D : \mathcal{Z} \to \mathcal{S}$ extracts the task-relevant state s from latent state $z \in \mathcal{Z}$. Combined with the oracle encoder in BMDPs, this enables recovery of the task-relevant state directly from observations: $s = D(q^{-1}(x))$. We define this composite function $\phi^* = D \circ q^{-1}$ as the **oracle encoder** of EX-BMDP.

2.2 Representation Learning in RL

In actor-critic methods (Konda & Tsitsiklis, 1999), representation learning is commonly used to handle complex MDPs such as EX-BMDPs. The idea is to learn an encoder that maps a raw observation to a representation, which is then shared by both actor and critic. Formally, an actor-critic algorithm employs an encoder $\phi : \mathcal{X} \to \Psi$, a (latent) actor $\pi_\theta : \Psi \to \Delta(\mathcal{A})$, and a (latent) critic $Q_\omega : \Psi \times \mathcal{A} \to \mathbb{R}$, where Ψ is the representation space. In this work, we focus on end-to-end actor-critic methods based on the soft actor-critic (**SAC**) algorithm (Haarnoja et al., 2018). These methods jointly optimize the encoder and actor-critic using the RL loss in SAC, denoted as $J_{\text{SAC}}(\phi, \theta, \omega)$.

Learning state representations solely from reward signals (i.e., RL loss) is challenging in complex tasks. To address this, various state abstraction frameworks and representation objectives have been proposed (see Ni et al. (2024) for a literature review). Among these, *model-irrelevance abstraction* (Li et al., 2006) defines two conditions for an effective encoder using *bisimulation relation* (Givan et al., 2003). The first condition, known as **reward prediction (RP)**[3], requires that the representation preserves reward information. The second condition, known as **self-prediction (ZP)**[4] (Ni et al., 2024), requires that the representation preserves latent dynamics information. Model-irrelevance abstraction thus defines compact yet informative encoders that retain sufficient information for optimal decision-making (Subramanian et al., 2022). By definition, the RP and ZP conditions hold when $\phi = \phi^*$ and $\Psi = \mathcal{S}$.[5] This implies that the oracle encoder ϕ^* serves as a model-irrelevance abstraction.

[3]Formally, in an EX-BMDP, RP condition is $\exists R_\kappa : \Psi \times \mathcal{A} \to \mathbb{R}$, s.t. $R(\phi^*(x), a) = R_\kappa(\phi(x), a), \forall x, a$.
[4]Formally, in an EX-BMDP, ZP condition is $\exists P_\nu : \Psi \times \mathcal{A} \to \Delta(\Psi)$, s.t. $P(\psi' \mid x, a) = P_\nu(\psi' \mid \phi(x), a), \forall x, a, \psi'$.
[5]In this case, $R_\kappa(s, a) = R(s, a)$ and $P_\nu(s' \mid s, a) = p(s' \mid s, a)$.

To learn a model-irrelevance abstraction, **DeepMDP** (Gelada et al., 2019) introduces **RP and ZP losses** to approximate the RP and ZP conditions, respectively. Given a data tuple (x, a, r, x'), these losses jointly optimizes the encoder ϕ, the reward model R_κ, and the latent transition model P_ν:

$$J_{\text{RP}}(\phi, \kappa) = (R_\kappa(\phi(x), a) - r)^2, \quad J_{\text{ZP}}(\phi, \nu) = -\log P_\nu(\bar{\phi}(x') \mid \phi(x), a), \tag{1}$$

where $\bar{\phi}$ detaches the encoder from gradient back-propagation. The overall objective $J_{\text{DeepMDP}}(\phi)$ for the encoder in DeepMDP combines SAC loss with RP and ZP losses (Eq. 1).

3 Conceptual Analysis on Behavioral Metrics Learning in RL

This section establishes a conceptual framework linking behavioral metrics to representations in deep RL (Sec. 3.1), and then summarizes how related work instantiates it (Sec. 3.2).

3.1 Isometric Embedding: Between Behavioral Metrics and Representation

We aim to find an encoder that maps noisy observations into a structured representation space, where distances reflect differences in rewards and transition dynamics smoothly. This representation should facilitate RL by ensuring that task-relevant variations are captured. A natural way to formalize this goal is through the concept of an *isometric embedding (isometry)*[6]:

Definition 1 (Isometric Embedding) *An encoder $\phi : \mathcal{X} \to \Psi$ is an* isometric embedding *if the distances in the original space $(\mathcal{X}, d_\mathcal{X})$ are preserved in the representation space (Ψ, d_Ψ). Formally,*

$$d_\mathcal{X}(x_1, x_2) = d_\Psi(\phi(x_1), \phi(x_2)), \quad \forall x_1, x_2 \in \mathcal{X}, \tag{2}$$

*where $d_\mathcal{X}$ is the **target metric** ("desired" metric) and d_Ψ is the **representational metric**. See Appendix Sec. B.1 for background on metric definitions.*

3.2 Design Choices in Metric Learning

Table 1: **Summary of key implementation choices for the benchmarked methods.**

Method	d_R	d_Ψ	d_T	Other Losses	Transition Model	Metric Loss	Normalization	Target Trick
SAC	—	—	—	—	—	—	—	—
DeepMDP	—	—	—	RP + ZP	Probabilistic	—	—	—
DBC	Huber	Huber	W_2 closed-form	RP + ZP	Probabilistic	MSE	—	—
RDBC	Huber	Huber	W_2 closed-form	RP + ZP	Deterministic	MSE	Max norm	—
MICo	Abs.	MICo Angular	Sample-based	—	—	Huber	—	✓
SimSR	Abs.	Cosine	Sample-based	ZP	Probabilistic Ensemble	Huber	L_2 norm	—
RAP	RAP	MICo Angular	W_2 closed-form	RP + ZP	Probabilistic	Huber	—	—

Def. 1 provides a general conceptual framework instantiated by several works in deep RL through distinct design choices. Rather than delving into theoretical implications, we focus on *practical implementations* reflected in their publicly available codebases, summarized in Table 1.

Choices of Target Metric $d_\mathcal{X}$. The target metric, which captures differences in rewards and transition dynamics, is typically formulated as (Castro et al., 2023): for $x_1, x_2 \in \mathcal{X}$,

$$d_\mathcal{X}(x_1, x_2) = c_R d_R(r_1, r_2) + c_T d_T(d_\mathcal{X})(P(\cdot \mid x_1), P(\cdot \mid x_2)), \tag{3}$$

which is inspired by bisimulation metrics (Ferns et al., 2004; 2011).[7]

- d_R, representing *immediate* state similarity, measures the closeness of sampled immediate rewards $r_1, r_2 \in \mathbb{R}$ based on x_1, x_2. Common choices include absolute difference $d_R(r_1, r_2) = |r_1 - r_2|$ ("Abs." in Table 1) (Zang et al., 2022; Castro et al., 2021), "Huber" distance[8] $d_R(r_1, r_2) = \frac{1}{2}(r_1 - r_2)^2 \mathbf{1}_{\{|r_1 - r_2| < 1\}} + \left(|r_1 - r_2| - \frac{1}{2}\right) \mathbf{1}_{\{|r_1 - r_2| \geq 1\}}$ (Huber in Table 1), or other specific forms (Chen & Pan, 2022).

[6] https://en.wikipedia.org/wiki/Isometry

[7] See Appendix Sec. B.2 for the full formal definition and discussion of the variants of bisimulation metrics.

[8] https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html

- d_T, a probabilistic measure of *long-term* state similarity, typically avoids expensive 1-Wasserstein computations in bisimulation metrics. Methods such as (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021; Chen & Pan, 2022) approximate transitions using a Gaussian transition model with a 2-Wasserstein metric. In contrast, Castro et al. (2021); Zang et al. (2022) rely on sample-based distance approximations.

Choices of Representational Metric d_Ψ. To approximate d_Ψ, methods employ either a Huber distance (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021) (a surrogate for L_p-distance) or an angular distance (Castro et al., 2021; Zang et al., 2022; Chen & Pan, 2022).

Metric Loss Function J_M. To approximate an isometric embedding, metric learning methods optimize this general objective:

$$J_M(\phi) = \ell(d_\Psi(\phi(x_1), \phi(x_2)) - d_\mathcal{X}(x_1, x_2)), \tag{4}$$

where ℓ can be Huber loss (Chen & Pan, 2022; Castro et al., 2021; Zang et al., 2022), or mean square error (MSE) (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021).

Self-prediction (ZP). As discussed, approximating $d_\mathcal{X}$ often requires a transition model, methods adopt distinct approaches: probabilistic models (Zhang et al., 2020; Castro et al., 2021), ensembles of probabilistic models (Zang et al., 2022), and deterministic models (Kemertas & Aumentado-Armstrong, 2021). MICo, in contrast, employs a sample-based target metric that is free of ZP.

Normalization. We discuss normalization in the representation space Ψ. RDBC (Kemertas & Aumentado-Armstrong, 2021) employs max normalization to enforce boundedness, leveraging prior knowledge of target metric constraints. SimSR (Zang et al., 2022) applies L_2 normalization to enforce unit-length representations. LayerNorm (Ba et al., 2016) is widely used in pixel-based encoder across all methods except SimSR. In the broader context of deep RL, normalization has been extensively studied for its benefits in stabilizing training and generalization (Li et al., 2023; Fujimoto et al., 2023; Elsayed et al., 2024; Gallici et al., 2024).

Target trick. MICo employs a target network $\bar{\phi}$ for encoding one observation in d_Ψ when approximating $d_\mathcal{X}$ to ensure learning stability. See Castro et al. (2021, Appendix C.2) for further details.

3.3 Candidate Methods

We present the design choices of methods to be benchmarked in our work in Table 1. Note that RDBC (Kemertas & Aumentado-Armstrong, 2021) incorporates additional components – such as intrinsic rewards and inverse dynamics – that enhance performance. However, since these elements are orthogonal to our study, they are not included in our implementation. For a fair comparison, our experiments employ a probabilistic transition model for all methods that require one.

4 Study Design: Does Metric Learning Help with Denoising?

The "denoising capability" of behavioral metric learning is often cited as a motivation in prior work (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021; Chen & Pan, 2022; Zang et al., 2022). However, most studies evaluate this *indirectly* by (1) combining metric learning with RL, (2) training only on grayscale natural video backgrounds, (3) testing on unseen videos in training, and (4) evaluating solely through return performance. This leaves a gap between motivation and actual denoising assessment.

This section bridges that gap with a systematic study design. First, we introduce a diverse range of noise settings from IID Gaussian noise and random projections to natural video backgrounds (Sec. 4.1), enabling an analysis of how noise difficulty impacts metric learning. Second, we separate the noise distributions during training and testing to examine denoising under both ID and OOD

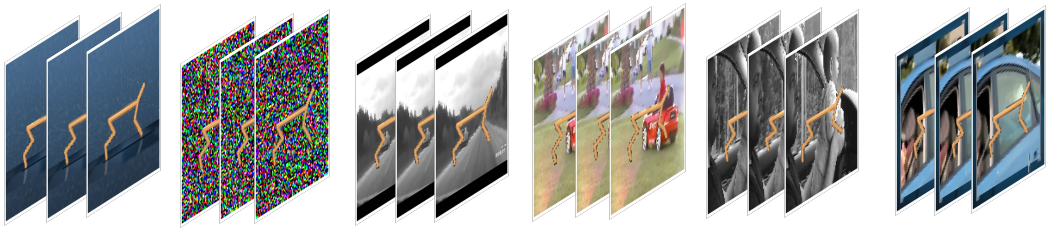

Figure 1: Examples of different noise settings in pixel-based domains (three consecutive timesteps each). **From left to right**: original clean image, IID Gaussian image, grayscale natural image, colored natural image, grayscale natural video, colored natural video. Three background images are different for video settings.

generalization settings (Sec. 4.2). Third, we introduce a direct evaluation measure, the *denoising factor* (Sec. 4.3). Finally, to disentangle metric learning from RL, we propose the *isolated metric estimation* setting, where metric learning affects only the encoder, not the RL agent (Sec. 4.4).

4.1 Noise Settings

We describe our noise settings using the EX-BMDP framework (Sec. 2.1), where observations follow $x \sim q(\cdot \mid z)$ with $z = (s, \xi)$.

IID Gaussian Noise. The task-irrelevant noise ξ_t is sampled independently at each timestep from an m-dimensional isotropic Gaussian, $\xi_t \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I})$. For state-based domains, the observation is exactly the latent state, i.e., $x_t = z_t$ with q as the identity mapping. We adjust the noise dimension m or noise std σ to modulate difficulty, whereas prior work (Kemertas & Aumentado-Armstrong, 2021; Ni et al., 2024) only varies m with a small σ. For pixel-based domains, noise is applied per pixel in the background and overlaid by the robot foreground's pixels, with q as a rendering function.

IID Gaussian Noise with Random Projection. This setting applies only to state-based domains where $s \in \mathbb{R}^n$. Before interaction with the MDP, we construct a full-rank square matrix $\mathbf{A} \in \mathbb{R}^{(n+m)\times(n+m)}$ with entries sampled as $A_{ij} \sim \mathcal{N}(\mu_A, \sigma_A^2)$. At each time step, we generate m-dimensional IID Gaussian noise $\xi_t \sim \mathcal{N}(0, \sigma^2\mathbf{I})$ and then apply a linear projection to obtain observation $x_t = \mathbf{A}z_t$ where $z_t = (s_t, \xi_t)$. Since $\mathbf{A}$ is full rank, s_t can be recovered from x_t using $\mathbf{A}^{-1}$. This setting is more challenging than IID Gaussian noise, as it linearly entangles s_t and ξ_t, with q as the linear projection.[9]

Natural Images. This setting applies only to pixel-based domains, replacing the clean background with a randomly selected natural image. As in the original environment, the background remains fixed during training. Images can be *grayscale* or *colored*, introducing different levels of visual complexity. In EX-BMDP notation, ξ_t is stationary and q is a rendering function.

Natural Videos. This setting also applies only to pixel-based domains, replacing the clean background with with natural video. The underlying noise $\xi_t \in \mathbb{N}$, representing the *frame index*, follows the update rule $\xi_t = (\xi_{t-1} + 1) \mod N$, where N is the total number of frames. These videos can be *grayscale* or *colored*, with the grayscale setting widely used in metric learning (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021; Zang et al., 2022; Chen & Pan, 2022).

4.2 Denoising Involves ID and OOD Generalization

In this work, "*denoising*" refers to a form of generalization that removes task-irrelevant noise from observations, enabling generalization to tasks with unseen noise. The evaluation settings differ based on whether the *noise distribution* remains unchanged or shifts between training and testing.

[9] Voelcker et al. (2024) similarly projects observations using a random binary matrix in discrete domains.

In-distribution (ID) Generalization Evaluation. The training and testing environments (EX-BMDPs) are identical, meaning the same noise distribution is applied in both phases. For example, IID Gaussian noise remains unchanged throughout training and testing.

Out-of-distribution (OOD) Generalization Evaluation. The training and testing EX-BMDPs share the same task-relevant parts (i.e., $p(s' \mid s, a)$, $p(s_0)$, $R(s, a)$) but differ in noise distributions (i.e., $p(\xi' \mid \xi)$, $p(\xi_0)$). For instance, natural videos from a training dataset are employed during training, while videos from a distinct test dataset are used during evaluation. This evaluation setup is widely used in metric learning (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021; Zang et al., 2022; Chen & Pan, 2022).

4.3 Quantifying Denoising via the Denoising Factor

We introduce the *denoising factor* (DF), a measure that quantifies an encoder ϕ's ability to filter out irrelevant details while retaining essential information.[10] It also provides insight into how the behavioral metrics are approximated, given that exact behavioral metrics are nearly inaccessible via fixed-point iteration in high-dimensional state or action spaces. To compute DF, we define a *positive score* and a *negative score* for an encoder ϕ. Inspired by triplet loss (Schroff et al., 2015) in contrastive learning, we compute these scores by selecting an observation x as an *anchor*, then constructing a *positive example* x^+ similar to x, and a *negative example* x^- dissimilar to x.

Definition 2 (Positive score) *The positive score of an encoder ϕ w.r.t. the metric d_Φ measures the average representational distance between anchors and their positive examples:*

$$\mathrm{Pos}_{d_\Phi}(\phi) := \mathbb{E}_{x\sim\rho_\pi(x),\xi^+\sim\rho(\xi^+),x^+\sim q(\cdot|\phi^*(x),\xi^+)}\big[d_\Phi(\phi(x),\phi(x^+))\big], \tag{5}$$

where $\rho_\pi(x)$ is the stationary state distribution under the policy π and $\rho(\xi^+)$ is a stationary noise distribution. The sampling $x^+ \sim q(\cdot \mid \phi^*(x), \xi^+)$ ensures that x^+ shares the same task-relevant state $s = \phi^*(x)$ but has different noise ξ^+.

In the temporally-independent noise setting, $\rho(\xi^+)$ matches the noise transition; in the natural-video setting, $\rho(\xi^+)$ is a uniform distribution over frame indices $\{0, 1, \ldots, N-1\}$.

Definition 3 (Negative score) *The negative score of an encoder ϕ w.r.t. the metric d_Φ measures the average representational distance between anchors and their negative examples:*

$$\mathrm{Neg}_{d_\Phi}(\phi) := \mathbb{E}_{x,x^-\overset{\text{iid}}{\sim}\rho_\pi}\big[d_\Phi(\phi(x),\phi(x^-))\big], \tag{6}$$

where x, x^- are IID samples from ρ_π.

Definition 4 (Denoising factor (DF)) *The denoising factor of an encoder ϕ w.r.t. the metric d_Φ is defined as the normalized difference between the negative and positive scores. As a result, a higher DF indicates better denoising ability.*

$$\mathrm{DF}_{d_\Phi}(\phi) := \frac{\mathrm{Neg}_{d_\Phi}(\phi) - \mathrm{Pos}_{d_\Phi}(\phi)}{\mathrm{Neg}_{d_\Phi}(\phi) + \mathrm{Pos}_{d_\Phi}(\phi)} \in [-1, 1]. \tag{7}$$

4.4 Decoupling Behavioral Metric Learning from RL

In many behavioral metric learning methods, the encoder ϕ is optimized via a combination of losses: the RL loss (e.g., $J_{\mathrm{SAC}}(\phi)$), the reward-prediction loss $J_{\mathrm{RP}}(\phi)$, the self-prediction loss $J_{\mathrm{ZP}}(\phi)$ (Eq. 1), and a metric loss $J_{\mathrm{M}}(\phi)$ (3). This coupling makes it difficult to isolate the direct impact of metric learning on representation quality. Moreover, denoising factor (DF, Def. 4) depends on both the encoder and the data distribution induced by RL agent. Policies that frequently revisit similar task-relevant states under varying noise may inflate DF, making it an unreliable measure of denoising. Due to the above reasons, we propose to evaluate behavioral metric learning algorithms in an *isolated metric estimation* setting.

[10] While the oracle encoder ϕ^* achieves perfect denoising, direct comparison is impossible as ϕ lacks access to $\mathcal{S}$.

Isolated Metric Estimation Setting. To isolate the effect of metric learning, we introduce an isolated *metric encoder* $\tilde{\phi}$ that is optimized solely via the metric loss $J_{\mathrm{M}}(\tilde{\phi})$, while the *agent encoder* ϕ is updated using the remaining training objectives (*e.g.*, $J_{\mathrm{SAC}}(\phi)$ or $J_{\mathrm{DeepMDP}}(\phi)$). In our experiments, regardless of the metric learning method, a SAC agent interacts with the environment and collect data for learning the metrics. This allows for a fair comparison of $\mathrm{DF}_{d_\Phi}(\tilde{\phi})$ across different metric learning methods. For methods that rely on self-prediction loss (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021; Zang et al., 2022), we learn an *isolated transition model* using $\tilde{\phi}$ while preventing gradient backpropagation to $\tilde{\phi}$ to ensure isolation.

5 Experiments

Experiment Organization. We first conduct a comprehensive evaluation of all the methods (Table 1) across **20 state-based** DeepMind Control (DMC) (Tassa et al., 2018; Tunyasuvunakool et al., 2020) tasks (listed in Table 5) and **14 pixel-based** DMC tasks (listed in Table 6). Evaluations are performed under various noise settings using ID generalization. This study covers a significantly larger task set than prior works. Our results (Sec. 5.1) provide a broad assessment of agent performance and task difficulty, as reflected by return variations. Based on these findings, we select a subset of representative tasks for fine-grained analysis (Sec. 5.2) to examine the impact of key design choices (Sec. 3.2). We further investigate the isolated metric evaluation setting (Sec. 4.4) in Sec. 5.3, and assess OOD generalization following prior work in Sec. 5.4.

Evaluation Protocol. We report the *mean episodic reward* rather than the IQM (Agarwal et al., 2021) to avoid ignoring tasks that are too easy or too challenging. For each run, the reported mean episodic reward, bounded within $[0, 1000]$ for all tasks, is the average of 10 evaluation points within a 1.95M-2.05M step window and aggregated over seeds. Figures and tables display 95% confidence intervals over tasks. For state-based environments, we use 12 random seeds per *configuration*, where a configuration is defined as a *(task, noise setting)* pair. For pixel-based experiments, we use 5 random seeds per configuration.

Approximation of Denoising Factor (Eq. 7). All observations collected in the evaluation stage are regarded as *anchors*. We sample 16 positive samples and negative samples for each anchor using the strategy shown in Sec. 4.3. For consistency, we report $\mathrm{DF}_{\|\cdot\|_2}(\phi)$.

5.1 Benchmarking Methods on Various Noise Settings

Settings. For state-based DMC tasks, we apply IID Gaussian noise, varying either (a) standard deviations $\sigma \in \{0.2, 1.0, 2.0, 4.0, 8.0\}$ (with a fixed noise dimension $m = 32$), or (b) noise dimensions $m \in \{2, 16, 32, 64, 128\}$ (with a fixed standard deviation $\sigma = 1.0$). For pixel-based DMC tasks, evaluation is conducted under **6 image background settings**: (1) clean background (the original pixel-based DMC setting), (2) grayscale natural images, (3) colored natural images, (4) grayscale natural videos, (5) colored natural videos, and (6) IID Gaussian noise (with $\sigma = 1.0$). ID generalization evaluation is conducted in this subsection. The aggregated reward and DF for settings (a), (b), and (1)-(6) are shown in Fig. 2 and Fig. 3, respectively. Per-task results are listed in Appendix Sec. E.

Implementation Details. For state-based tasks, the encoder is a three-layered MLP, as used by SAC (Haarnoja et al., 2018) and RDBC (Kemertas & Aumentado-Armstrong, 2021). For pixel-based tasks, the encoder is a CNN followed by LayerNorm (Ba et al., 2016), as used by SAC-AE (Yarats et al., 2021b). All the compared methods are implemented based on SAC. For a fair comparison, we adopt an identical probabilistic latent transition models and reward models used in DBC and RDBC, although some methods, such as SimSR (Zang et al., 2022), propose using an ensemble of latent transition models. We follow the exact hyperparameters specific to each metric learning method. Please see Appendix Sec. D for further details.

Benchmarking Findings. We summarize the key findings from Fig. 2 and Fig. 3.

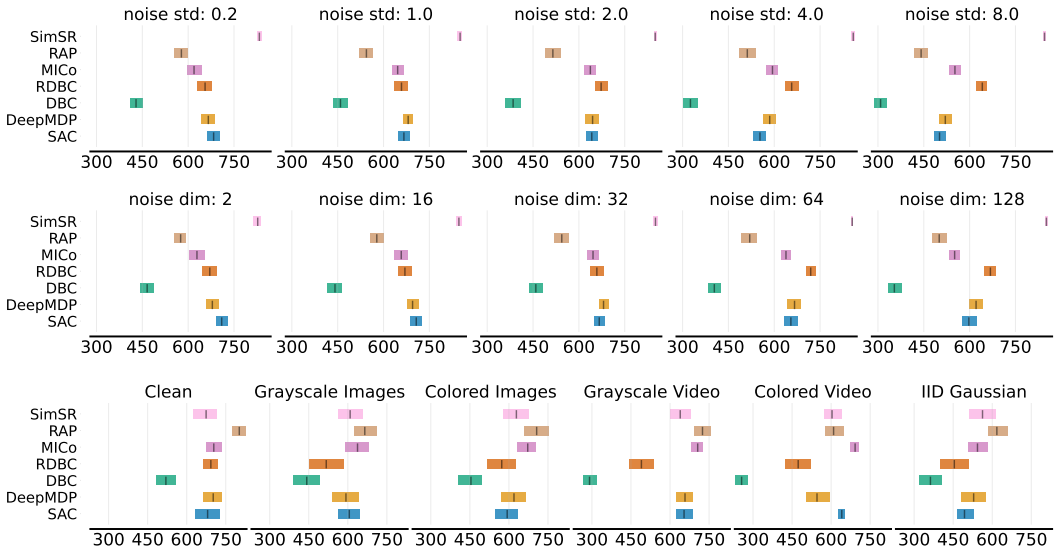

Figure 2: **Benchmarking results: performance of seven methods across diverse noise settings**, aggregating episodic rewards from **20 state-based** (first two rows) and **14 pixel-based** tasks (last row). "Noise std" denotes the IID Gaussian noise's standard deviation σ, while "noise dim" denotes its dimension m. Bars show 95% CI.

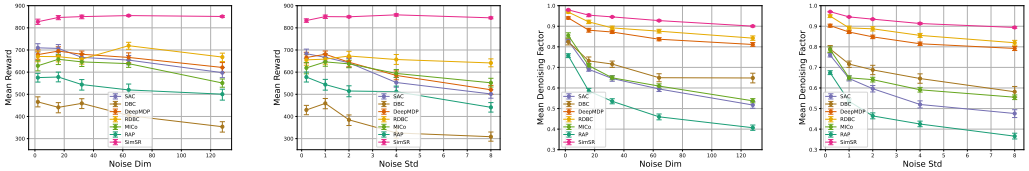

Figure 3: **Benchmarking results: reward (left) and denoising factor (right)** of seven methods to IID Gaussian noise dimension (Noise Dim) and standard deviation (Noise Std). Each point is aggregated by 20 state-based tasks in Table 5.

- **SimSR** consistently achieves the highest performance in most state-based tasks, excelling in both return and denoising factor. **RAP** performs best in most pixel-based tasks but suffers a moderate performance drop in state-based tasks. Interestingly, both SimSR and RAP were evaluated only in pixel-based domains in their papers, making our state-based findings novel.
- **SAC and DeepMDP**, as fundamental metric learning baselines, deliver decent performance on both pixel-based and state-based tasks. However, they are often overlooked in prior work.
- In state-based domains, all methods are generally more robust to increased noise *dimensions* (when $\sigma = 1.0$), as commonly used in prior work, than to increased *standard deviation* (when $m = 32$), as shown in Fig. 3.
- In pixel-based domains, *grayscale natural video*, widely used in prior work, is not significantly harder than clean background setting (e.g., for SAC and DeepMDP). Surprisingly, the *IID Gaussian noise* setting is the most challenging, warranting further study.
- Different algorithms excel in different tasks (Table 5, Table 6), e.g., RAP in reacher/easy, MICo in point_mass/easy (Fig. 23). Broad task coverage is essential to ensure generalizable insights.
- Adding objectives trades-off computation efficiency. As shown in Table 2, the time cost of optimizing of a metric loss is close to optimizing a ZP loss by comparing MICo with DeepMDP.

5.2 What Matters in Metric and Representation Learning?

To identify key factors in metric learning in state-based domains, we conduct case studies on the design choices outlined in Sec. 3.2. We select *three medium-to-hard DMC tasks*, finger/turn_easy, figure/turn_hard, quadruped/run for detailed analysis. First, a notable difference between our default encoder implementations for state-based and pixel-based tasks is the inclusion of normalization, which may significantly impact benchmarking outcomes. SimSR, the best-performing algorithm in state-based environments, employs L_2 normalization in the representation space and discusses its

Table 2: **Relative time spent on model updates** on NVIDIA L40S GPUs under the same task (walker/walk, with $\mathcal{S} = \mathbb{R}^{24}$ and $\Xi = \mathbb{R}^{32}$). Values represent the multiple of SAC's updating time. Key hyperparameters affecting the speed are set identically to Table 7.

	SAC	DeepMDP	DBC	RDBC	MICo	RAP	SimSR
Pixel-based	1.00	1.44	2.03	2.12	1.53	2.20	1.75
State-based	1.00	1.42	1.76	1.95	1.39	2.08	1.68

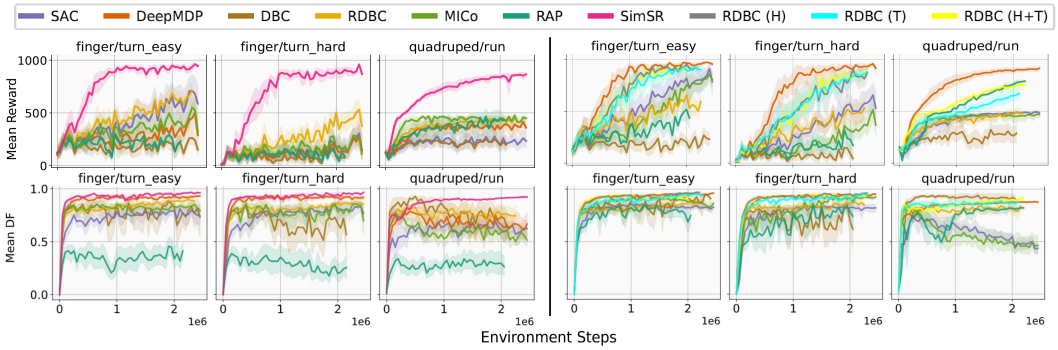

Figure 4: Case study on three DMC state-based tasks (IID Gaussian noise, noise dim=32, noise std=8.0), examining the effects of **including LayerNorm** (left vs. right three columns), **applying the target trick** (RDBC (T)), and **using Huber loss** (RDBC (H)) instead of MSE as the metric loss.

effectiveness (Zang et al., 2022). This inspires us to examine whether normalization benefits *other* metric learning methods. Second, several techniques used by the best-performing methods merit further analysis. For instance, SimSR, RAP, and MICo (which excels in *colored natural video* settings) utilize Huber metric loss instead of MSE, while MICo incorporates the target trick (Sec. 3.2). Third, we investigate the performance of methods *with LayerNorm* in a challenging setting: *IID Gaussian noise with random projection* (Sec. 4.1) with $\sigma \in \{0.2, 2.0, 4.0, 8.0\}$ (with a fixed noise dimension $m = 32$), shown in Fig. 6. Important findings from Fig. 4 to Fig. 6 is as follows:

- **Most methods benefit from LayerNorm in the representation space**, improving both reward and DF (Fig. 4). Notably, **DeepMDP with LayerNorm performs comparably to SimSR**.[11] For RDBC, using Huber loss for the metric and using the target trick enhance performance (Fig. 4).[12]
- **ZP loss is crucial for SimSR's success** in noisy state-based tasks (Fig. 5).
- A significant performance drop occurs for all agents when increasing the noise standard deviation in **IID Gaussian with random projection** setting (Fig. 6), even with normalization applied. Nevertheless, DeepMDP and SimSR remain relatively robust to the noise.

5.3 Isolated Metric Evaluation Setting: Does Learned Metrics Denoise?

We further analyze the proposed setting in Sec. 4.4 in both state-based (shown in Fig. 7) and pixel-based settings (shown from Fig. 25 to Fig. 29). We observe that:

- Generally, metrics learned in isolation achieve some denoising but underperform compared to those co-learned with ZP and critic losses or even with ZP and critic losses alone in DeepMDP (first two rows of Fig. 7).
- LayerNorm also works well with isolated metric estimation (last two rows of Fig. 7).
- MICo's DF remains relatively low, which aligns with its theoretical property that the metric for positive samples is non-zero (Fig. 7), as MICo does not enforce zero self-distance.

[11] Our additional experiments reveal that removing LayerNorm in pixel-based environments causes a substantial performance drop across all methods, highlighting the critical role of normalization.

[12] We observe that runs with superior design choices exhibit much more stable representation norms and gradient norms for both the critic loss and metric loss. Thus, the performance gains shown in Fig. 4 are likely due to improved numerical stability in extrapolating the metrics and Q values during generalization.

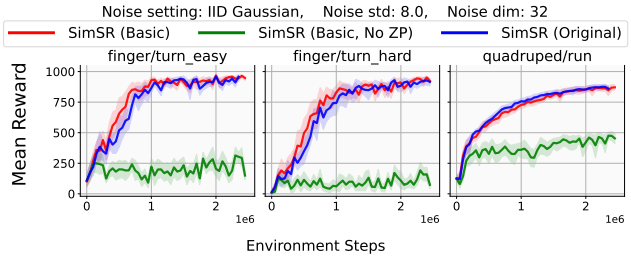

Figure 5: **Ablation study on ZP loss on SimSR.** "SimSR (Original)" is the configuration benchmarked in Sec. 5.1, where ZP is integral to the metric estimation. Therefore, we resort to "SimSR (Basic)" setting (Theorem 2, Zang et al. (2022)), where the ZP component is independent from the metric estimation, and "SimSR (Basic, No ZP)" is the setting that ZP is detached from SimSR (Basic). This ablation highlights the impact of detaching ZP on the overall performance.

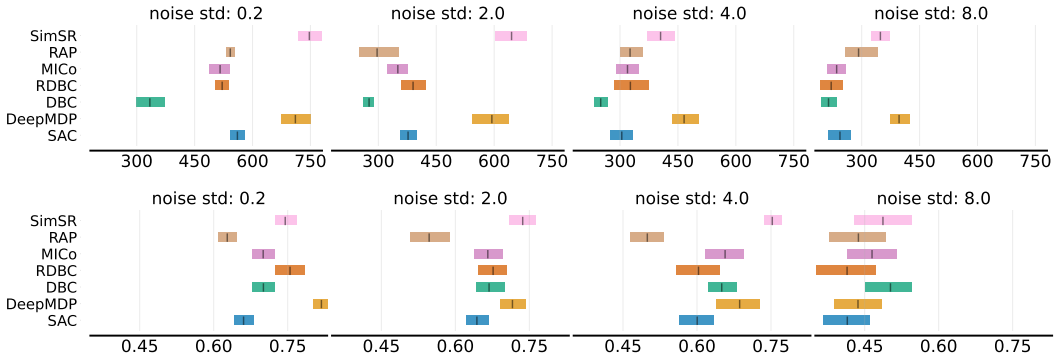

Figure 6: Aggregated reward (top row) and DF (bottom row) of seven agents on various **IID Gaussian with random projection** settings in the 3 selected state-based tasks.

5.4 OOD Generalization

While prior work has focused on OOD generalization in pixel-based settings, we extend this analysis by evaluating all **14 pixel-based tasks**. The "grayscale video" setting (and similarly for other settings) in Fig. 8 denotes using grayscale videos for both training and evaluation, with distinct video samples in each phase. Takeaways in Fig. 8 and Fig. 9 are as follows:

- Comparing Fig. 8 (OOD) with Fig. 2 (ID), all methods struggle to generalize in both grayscale and colored image settings. Unlike video backgrounds, which provide temporal variation, static images lack diverse cues, making adaptation to unseen backgrounds more challenging.
- Even with OOD generalization evaluation, SAC and DeepMDP remain competitive baselines (Fig. 8).
- OOD generalization is more challenging in the *colored* video setting than in the *grayscale* video setting (Fig. 9). Surprisingly, even baselines like SAC exhibit a low reward gap, questioning the necessity of incorporating a metric loss in the widely-used grayscale setting.

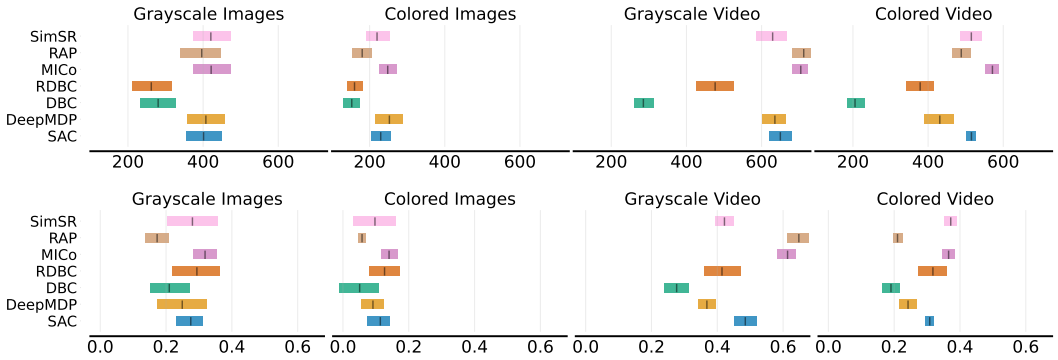

Figure 8: Aggregated reward (top row) and DF (bottom row) of seven agents on various noise settings in 14 pixel-based tasks in Table 6 with **OOD generalization** evaluation.

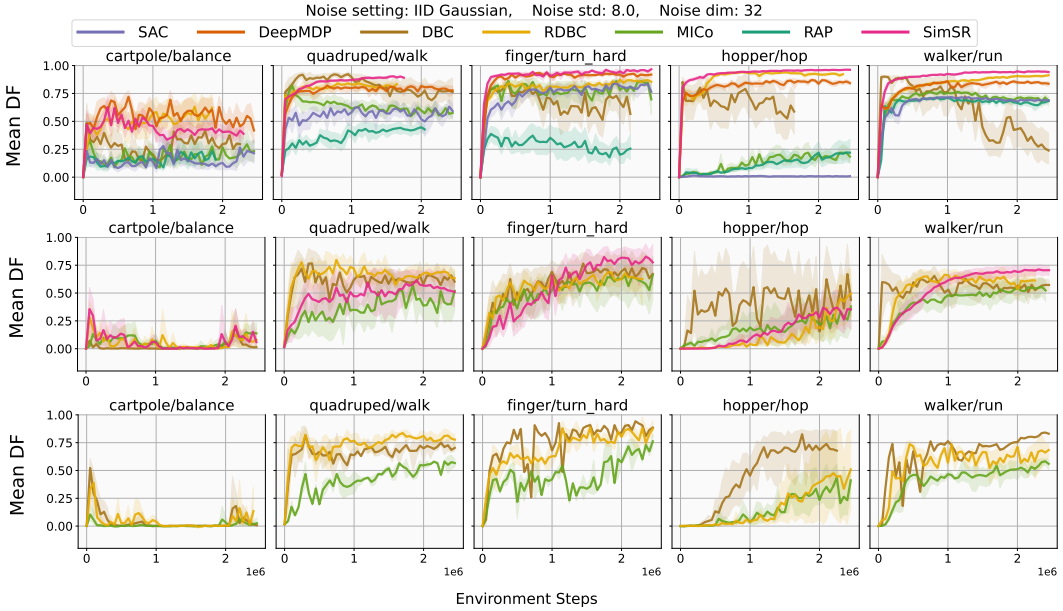

Figure 7: **Top row**: DF for the agent encoder ϕ (co-trained with RL in Sec. 5.1) without LayerNorm. **Mid row**: DF for the isolated metric encoder $\tilde{\phi}$ without LayerNorm. **Bottom row**: DF for $\tilde{\phi}$ with LayerNorm. All use ID generalization evaluation.

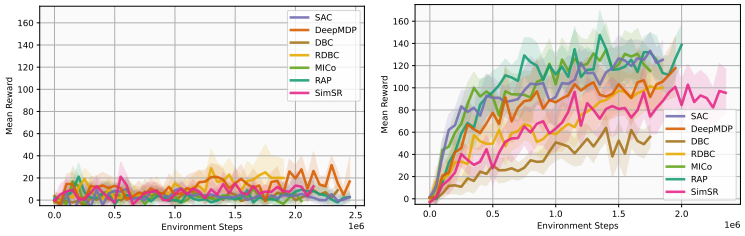

Figure 9: **Reward gap** (performance in ID evaluation minus OOD evaluation) in the grayscale video setting (left) and the colored video setting (right), aggregated on 14 pixel-based tasks in Table 6.

6 Conclusion

Takeaways. Based on our empirical studies on behavioral metric learning in deep RL, we highlight the following key insights for RL researchers:

1. To gain a clearer understanding of RL algorithms, initial evaluations should be conducted on simple, controlled environments (e.g., varying Gaussian noise std, pixel-based Gaussian noise).
2. Claims and motivations for metric learning should be supported by rigorous evaluation, including measures such as the denoising factor and comparisons between ID and OOD generalization.
3. Self-prediction loss and LayerNorm are critical design choices that significantly impact metric and representation learning.
4. The benefits of metric learning diminish when key design choices, such as self-prediction loss and LayerNorm, are integrated into SAC. This calls for a deeper investigation into when and how metric learning provides unique advantages beyond these existing techniques.

Future work. Our study focuses on continuous control; future work should explore discrete domains and real-world tasks. The relationship between denoising and return performance remains unclear, requiring further investigation. Additionally, improved benchmarks are needed to better isolate the effects of metric learning.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 10069–10076, 2020.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. Mico: Improved representations via sampling-based state similarity for markov decision processes. *Advances in Neural Information Processing Systems*, 34:30113–30126, 2021.

Pablo Samuel Castro, Tyler Kastner, Prakash Panangaden, and Mark Rowland. A kernel perspective on behavioural metrics for markov decision processes. *arXiv preprint arXiv:2310.19804*, 2023.

Jianda Chen and Sinno Pan. Learning representations via a robust behavioral metric for deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:36654–36666, 2022.

Edmund M Clarke. Model checking. In *Foundations of Software Technology and Theoretical Computer Science: 17th Conference Kharagpur, India, December 18–20, 1997 Proceedings 17*, pp. 54–56. Springer, 1997.

Simon Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudik, and John Langford. Provably efficient rl with rich observations via latent state decoding. In *International Conference on Machine Learning*, pp. 1665–1674. PMLR, 2019.

Yonathan Efroni, Dipendra Misra, Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Provable rl with exogenous distractors via multistep inverse dynamics. *arXiv preprint arXiv:2110.08847*, 2021.

Mohamed Elsayed, Gautham Vasan, and A Rupam Mahmood. Streaming deep reinforcement learning finally works. *arXiv preprint arXiv:2410.14606*, 2024.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pp. 162–169, 2004.

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

Scott Fujimoto, Wei-Di Chang, Edward Smith, Shixiang Shane Gu, Doina Precup, and David Meger. For sale: State-action representation learning for deep reinforcement learning. *Advances in neural information processing systems*, 36:61573–61624, 2023.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. *arXiv preprint arXiv:2407.04811*, 2024.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International conference on machine learning*, pp. 2170–2179. PMLR, 2019.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial intelligence*, 147(1-2):163–223, 2003.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Riashat Islam, Manan Tomar, Alex Lamb, Yonathan Efroni, Hongyu Zang, Aniket Didolkar, Dipendra Misra, Xin Li, Harm Van Seijen, Remi Tachet des Combes, et al. Agent-controller representations: Principled offline rl with rich exogenous information. *arXiv preprint arXiv:2211.00164*, 2022.

Mete Kemertas and Tristan Aumentado-Armstrong. Towards robust bisimulation metric learning. *Advances in Neural Information Processing Systems*, 34:4764–4777, 2021.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

George Konidaris. On the necessity of abstraction. *Current opinion in behavioral sciences*, 29:1–7, 2019.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006.

Lu Li, Jiafei Lyu, Guozheng Ma, Zilin Wang, Zhenjie Yang, Xiu Li, and Zhiheng Li. Normalization enhances generalization in visual reinforcement learning. *arXiv preprint arXiv:2306.00656*, 2023.

Xiang Li, Jinghuan Shang, Srijan Das, and Michael Ryoo. Does self-supervised learning really improve reinforcement learning from pixels? *Advances in Neural Information Processing Systems*, 35:30865–30881, 2022.

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

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Satinder Singh, Tommi Jaakkola, and Michael Jordan. Reinforcement learning with soft state aggregation. *Advances in neural information processing systems*, 7, 1994.

Jayakumar Subramanian, Amit Sinha, Raihan Seraj, and Aditya Mahajan. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *Journal of Machine Learning Research*, 23(12):1–83, 2022.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Manan Tomar, Utkarsh A Mishra, Amy Zhang, and Matthew E Taylor. Learning representations for pixel-based control: What matters and why? *arXiv preprint arXiv:2111.07775*, 2021.

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.

Claas Voelcker, Tyler Kastner, Igor Gilitschenski, and Amir-massoud Farahmand. When does self-prediction help? understanding auxiliary tasks in reinforcement learning. *arXiv preprint arXiv:2406.17718*, 2024.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021a.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Proceedings of the aaai conference on artificial intelligence*, volume 35, pp. 10674–10681, 2021b.

Hongyu Zang, Xin Li, and Mingzhong Wang. Simsr: Simple distance-based state representations for deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pp. 8997–9005, 2022.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Notation

Table 3 and Table 4 show the glossary used in this paper.

Table 3: Glossary of notations in EX-BMDP (Sec. 2.1). The top section lists symbols related to the latent states, while the bottom section defines symbols related to grounded observations.

Symbol	Description
$z = (s, \xi) \in \mathcal{Z}$	Environment's latent state
$p(z' \mid z, a)$	Latent state transition
$s \in \mathcal{S}$	Task-relevant state
$\xi \in \Xi$	Task-irrelevant noise
$a \in \mathcal{A}$	Action
$R(s, a)$	Latent reward function
$r \in \mathbb{R}$	Reward
$\gamma \in [0, 1)$	Discount factor
$p(s' \mid s, a)$	Task-relevant state transition
$p(\xi' \mid \xi)$	Task-irrelevant noise transition
$x \in \mathcal{X}$	Observation
$q(x \mid z)$	Emission function
$q^{-1} : \mathcal{X} \to \mathcal{Z}$	Oracle encoder to $\mathcal{Z}$
$\phi^* : \mathcal{X} \to \mathcal{S}$	Oracle encoder (to $\mathcal{S}$)
$\mathcal{R}(x, a)$	Grounded reward function
$\mathcal{P}(x' \mid x, a)$	Grounded transition function

Table 4: Glossary of notations in RL agents.

Symbol	Description
$\psi \in \Psi$	Agent's representation
$\phi : \mathcal{X} \to \Psi$	Agent's encoder
$\pi_\theta : \Psi \to \Delta(\mathcal{A})$	(Latent) Actor
$Q_\omega : \Psi \times \mathcal{A} \to \mathbb{R}$	(Latent) Critic
$R_\kappa : \Psi \times \mathcal{A} \to \mathbb{R}$	(Latent) Reward model
$P_\nu : \Psi \times \mathcal{A} \to \Delta(\Psi)$	(Latent) Transition model
$\tilde{\phi} : \mathcal{X} \to \Psi$	(Isolated) Metric encoder

B Background on Metrics

B.1 Metric, Pseudometric, and Diffuse Metric

Metric. A function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is called a *metric* on the space $\mathcal{X}$ if for all $x, y, z \in \mathcal{X}$:

$$
\begin{aligned}
&(1) \quad d(x, y) = 0 \iff x = y, \\
&(2) \quad d(x, y) = d(y, x), \\
&(3) \quad d(x, z) \leq d(x, y) + d(y, z).
\end{aligned}
$$

Pseudometric[13]**.** A function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a *pseudometric* if it satisfies (2) and (3) above, and for (1) we only require $d(x, x) = 0$ for all $x \in \mathcal{X}$ (i.e., $d(x, y) = 0$ does not imply $x = y$).

Diffuse Metric (Definition 4.9 in Castro et al. (2021)). A function $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ is a *diffuse metric* if it satisfies properties (2) and (3) above, and for (1) we only require $d(x, y) \geq 0$. That is, we do not demand $d(x, x) = 0$ or that $d(x, y) = 0 \iff x = y$.

B.2 Definitions of Various Behavioral Metrics

In this section, we discuss the behavioral metrics for an EX-BMDP (Sec. 2.1). From the observation space $\mathcal{X}$, the grounded transition function is defined as $\mathcal{P}(x' \mid x, a) = \sum_{z' \in \mathcal{Z}} q(x' \mid z') p(z' \mid$

[13] Sometimes termed "semimetric" (Ferns et al., 2004).

$q^{-1}(x), a)$ and the grounded reward function as $\mathcal{R}(x, a) = R(\phi^*(x), a)$. Let $x_i, x_j \in \mathcal{X}$ be two arbitrary observations.

Bisimulation metric is a relaxation of bisimulation relation (Givan et al., 2003) by allowing a smooth variation based on differences in the reward function and transition dynamics. Originally rooted in the concept of identical future reward sequences, it is closely linked to model checking (Clarke, 1997). The bisimulation metric thus quantifies the behavioral similarity between two states and is formally defined as follows:

Definition 5 (Bisimulation metric $d^\sim$ (Ferns et al., 2004; 2011)) *There exists a unique pseudo-metric $d^\sim : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, called the bisimulation metric (BSM)[14], defined as:*

$$d^\sim(x_i, x_j) := \max_{a \in \mathcal{A}} \left(c_R |\mathcal{R}(x_i, a) - \mathcal{R}(x_j, a)| + c_T \mathcal{W}_1(d^\sim)(\mathcal{P}(\cdot \mid x_i, a), \mathcal{P}(\cdot \mid x_j, a))\right), \tag{8}$$

where 1-Wasserstein distance $\mathcal{W}_1(d^\sim)(P, Q) = \inf_\delta \mathbb{E}_{(x'_i, x'_j) \sim \delta}\left[d^\sim(x'_i, x'_j)\right]$, the coupling δ satifies $\sum_{x'_j} \delta(x'_i, x'_j) = P(x'_i)$, and $\sum_{x'_i} \delta(x'_i, x'_j) = Q(x'_j)$, c_R and c_T are coefficients for short-term and long-term behavioral differences.

In practice, applying the $\max$ operator over actions is intractable in continuous action spaces and pessimistically accounts for behavioral similarity across all actions, including those leading to low rewards. Policy-dependent bisimulation metrics (Castro, 2020) address this limitation by restricting behavioral similarity to the current policy, eliminating the need to evaluate all actions.

Definition 6 (Policy-dependent bisimulation metric d^π (Castro, 2020)) *Given a policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$, there exists a unique pseudometric $d^\pi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, called a policy-dependent bisimulation metric (PBSM), defined as:*

$$d^\pi(x_i, x_j) := c_R |\mathcal{R}^\pi(x_i) - \mathcal{R}^\pi(x_j)| + c_T \mathcal{W}_1(d^\pi)(\mathcal{P}^\pi(\cdot \mid x_i), \mathcal{P}^\pi(\cdot \mid x_j)), \tag{9}$$

where $\mathcal{R}^\pi(x) := \mathbb{E}_{a \sim \pi}[\mathcal{R}(x, a)]$ and $\mathcal{P}^\pi(\cdot \mid x) := \mathbb{E}_{a \sim \pi}[\mathcal{P}^\pi(\cdot \mid x, a)]$ are policy-dependent reward and transition, respectively.

To approximate the 1-Wasserstein distance in PBSM, DBC (Zhang et al., 2020) assumes a Gaussian transition kernel and uses 2-Wasserstein distance which has a close-form solution under such assumption. To further circumvent the costly computation of the 1-Wasserstein distance, Castro et al. (2021) proposes MICo distance.

Definition 7 (MICo distance u^π (Castro et al., 2021)) *Given a policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$, there exists a unique diffuse metric $u^\pi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, called MICo distance:*

$$u^\pi(x_i, x_j) := c_R |\mathcal{R}^\pi(x_i) - \mathcal{R}^\pi(x_j)| + c_T \mathbb{E}_{x'_i \sim \mathcal{P}^\pi(\cdot|x_i), x'_j \sim \mathcal{P}^\pi(\cdot|x_j)}\left[u^\pi(x'_i, x'_j)\right]. \tag{10}$$

Based on MICo, several improvements are made by SimSR (Zang et al., 2022) and RAP (Chen & Pan, 2022).

Definition 8 (SimSR distance (Zang et al., 2022)) *Given a policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$, there exists a unique distance $u^\pi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, called the Simple State Representation (SimSR) distance:*

$$u^\pi(x_i, x_j) \;:=\; c_R \left|\mathcal{R}^\pi(x_i) - \mathcal{R}^\pi(x_j)\right| \;+\; c_T\, \mathbb{E}_{x'_i \sim \widehat{\mathcal{P}}^\pi(\cdot|x_i),\; x'_j \sim \widehat{\mathcal{P}}^\pi(\cdot|x_j)}\left[u^\pi(x'_i,\, x'_j)\right], \tag{11}$$

where $u^\pi(x_i, x_j) = \cos(\phi(x_i), \phi(x_j))$ is the cosine distance and $\widehat{\mathcal{P}}^\pi$ is an approximated transition dynamics model.

Definition 9 (RAP distance (Chen & Pan, 2022)) *Given a policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$, there exists a unique distance $u^\pi : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, called the Robust Approximate (RAP) distance:*

$$u^\pi(x_i, x_j) \;:=\; c_R \left|\mathcal{R}^\pi(x_i) - \mathcal{R}^\pi(x_j)\right| \;+\; c_T\, \mathbb{E}_{a_i \sim \pi,\, a_j \sim \pi}\Big[\, u^\pi\Big(\, \mathbb{E}_{x'_i \sim \hat{\mathcal{P}}(\cdot|x_i, a_i)}[\, x'_i\,],\, \mathbb{E}_{x'_j \sim \hat{\mathcal{P}}(\cdot|x_j, a_j)}[\, x'_j\,]\,\Big)\Big]. \tag{12}$$

[14] As noted by Ferns et al. (2004), BSM relates to the largest bisimulation relation, $\sim$. For brevity, we simplify the original definition that uses the fixed-point of an operator and omit the proof for existence of such a fixed-point.

C Difficulty Levels of the Tasks

Difficulty levels for each task in state-based DMC are provided in Table 5. These levels are based on the average reward across all compared methods. In comparison with the difficulty assignment in pixel-based DMC within DrQ-v2 (Yarats et al., 2021a, Table 1), most tasks labeled as easy in our evaluation are also easy in DrQ-v2, and most tasks labeled as medium or hard in our evaluation are medium in DrQ-v2. However, finger spin and pendulum swingup shift from easy to medium, and hopper stand from easy to hard.

Table 5: **Difficulty levels for 20 state-based DMC tasks, as determined by the compared methods (Table 1).** "Avg Reward" stands for the average reward across all IID Gaussian noise settings in Sec. 5.1. For each run, the reported reward is the average of 10 evaluation points collected around 2M steps. "Max (Min) Reward" denotes the best (worst) agent's average reward over 12 runs, while "Max/Min" is the ratio of the best to worst performance, indicating a task's ability to discriminate between agent performances.

Task		Avg Reward	Max Reward	Min Reward	Max/Min	Difficulty
ball_in_cup	catch	934.8	977.4	841.7	1.2	Easy
cartpole	balance	919.4	997.3	791.2	1.3	Easy
cartpole	balance_sparse	877.7	983.6	772.3	1.3	Easy
walker	stand	834.6	979.0	437.8	2.2	Easy
cartpole	swingup	818.1	874.1	707.6	1.2	Easy
walker	walk	805.7	961.9	382.4	2.5	Easy
reacher	easy	740.1	955.1	453.0	2.1	Medium
finger	spin	728.8	923.6	498.5	1.9	Medium
quadruped	walk	703.1	948.9	245.5	3.9	Medium
cartpole	swingup_sparse	647.3	839.1	531.9	1.6	Medium
reacher	hard	641.1	853.0	340.3	2.5	Medium
finger	turn_easy	587.8	926.5	207.7	4.5	Medium
walker	run	545.8	776.1	117.4	6.6	Medium
cheetah	run	533.4	859.0	129.8	6.6	Medium
pendulum	swingup	514.3	824.5	247.2	3.3	Medium
quadruped	run	460.7	864.3	199.0	4.3	Hard
finger	turn_hard	435.6	893.0	102.6	8.7	Hard
hopper	stand	261.9	878.4	22.3	39.3	Hard
acrobot	swingup	75.7	246.1	11.2	22.0	Hard
hopper	hop	64.7	243.4	1.5	162.4	Hard

Table 6: **Difficulty levels for 14 pixel-based DMC tasks, as determined by the compared methods (Table 1).** "Avg Reward" stands for the average reward across clean background, natural video (colored and grayscale), natural image (colored and grayscale), and IID Gaussian noise settings described in Sec. 5.1. For each run, the reported reward is the average of 10 evaluation points collected around 2M steps. "Max (Min) Reward" denotes the best (worst) agent's average reward over 5 runs, while "Max/Min" is the ratio of the best to worst performance, indicating a task's ability to discriminate between agent performances.

Task		Avg Reward	Max Reward	Min Reward	Max/Min	Difficulty
cartpole	balance	949.3	986.7	905.5	1.1	Easy
cartpole	balance_sparse	915.3	999.4	804.6	1.2	Easy
walker	stand	887.7	959.1	633.7	1.5	Easy
finger	spin	815.2	909.5	426.4	2.1	Easy
cartpole	swingup	765.0	853.2	551.1	1.5	Medium
ball_in_cup	catch	719.2	887.8	263.4	3.4	Medium
walker	walk	718.9	909.1	360.8	2.5	Medium
point_mass	easy	421.5	558.6	256.1	2.2	Medium
cartpole	swingup_sparse	409.6	680.4	57.8	11.8	Medium
reacher	easy	336.8	949.1	113.0	8.4	Hard
pendulum	swingup	313.2	468.9	9.3	50.5	Hard
cheetah	run	299.4	411.0	144.7	2.8	Hard
walker	run	285.5	441.9	77.3	5.7	Hard
hopper	hop	73.6	122.5	5.6	22.0	Hard

D Hyperparameters

We align our hyperparameter settings with the agents we benchmark, referring to their open-source codebases and reported values in papers. Table 7 shows the general hyperparameter setting for most agents. For MICo (Castro et al., 2021)[15], β in MICo distance parametrization is set to 0.1. For RAP (Chen & Pan, 2022)[16], we use the same hyperparameter setting in their open-source code, where actor, critic, and encoder learning rate is set to 5×10^{-4}, β in MICo distance parametrization is set to 10^{-6}, RP and ZP loss coefficients is set to 10^{-4}, and encoder feature dimensionality is set to 100. For action repeat, we set it to 4 for most tasks, to 8 for cartpole (swingup, swingup_sparse), and to 2 for finger spin and walker (walk, run, stand) following the convention (Yarats et al., 2021a; Zang et al., 2022; Chen & Pan, 2022).

[15] https://github.com/google-research/google-research/tree/master/mico
[16] https://github.com/jianda-chen/RAP_distance

Table 7: **Hyperparameter settings** for most agents we benchmarked.

Hyperparameter Name	**Value**
Replay buffer capacity	1×10^6
Replay ratio	0.2
Batch size	128
Discount γ	0.99
Optimizer	Adam
Encoder feature dimensionality	50
Hidden units in neural networks	256
Paralleled environments	10
Critic learning rate	1×10^{-3}
Critic target update frequency	2
Critic Q-function soft-update rate τ_Q	0.01
Actor learning rate	1×10^{-3}
Actor update frequency	2
Actor log stddev bounds	$[-10, 2]$
Encoder learning rate	1×10^{-3}
Encoder soft-update rate τ_ϕ	0.05
Reward model and transition model's learning rate	1×10^{-3}
Reward model and transition model's weight decay	1×10^{-7}
SAC temperature learning rate	1×10^{-4}
SAC initial temperature	0.1
Metric loss weight λ_M	0.5
Metric reward coefficient c_R	1
Metric transition coefficient c_T	0.99
Image size	$84 \times 84 \times 3$
Frame stack	3
Distracting video frames N (Per paralleled environment)	1000

E Additional Experiment Results

E.1 Per-task Result for State-based DMC with IID Gaussian Noise Settings

In the figures and tables shown in this section, the tasks are sorted by their difficulty levels shown in Table 5 and 6.

Table 8: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=0.2, noise dim=32.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
ball_in_cup	catch	958.0 ± 3.9	971.0 ± 4.0	883.5 ± 114.0	975.0 ± 1.8	974.2 ± 1.4	966.6 ± 8.2	**978.3** ± 0.9
cartpole	balance	912.7 ± 24.6	836.8 ± 155.5	870.9 ± 172.0	988.7 ± 7.4	968.8 ± 16.7	967.6 ± 9.9	**995.0** ± 5.9
cartpole	balance_sparse	790.2 ± 143.2	579.4 ± 282.6	760.3 ± 193.4	624.0 ± 217.8	815.9 ± 182.7	882.7 ± 167.7	**991.9** ± 17.8
walker	stand	621.9 ± 164.8	929.3 ± 30.3	842.7 ± 65.6	974.6 ± 5.0	959.2 ± 28.5	846.2 ± 90.2	**977.4** ± 4.3
cartpole	swingup	776.0 ± 95.9	790.1 ± 143.1	872.7 ± 2.3	582.5 ± 203.8	867.0 ± 3.9	839.4 ± 13.0	**876.6** ± 5.2
walker	walk	443.1 ± 133.0	819.2 ± 62.2	869.8 ± 63.7	**963.8** ± 3.9	947.2 ± 6.4	780.6 ± 110.7	961.9 ± 7.0
reacher	easy	474.1 ± 219.0	501.3 ± 262.0	768.0 ± 169.8	556.0 ± 221.6	861.5 ± 146.6	831.6 ± 153.3	**957.0** ± 16.1
finger	spin	813.5 ± 36.1	837.9 ± 20.5	629.6 ± 152.9	905.8 ± 43.3	951.4 ± 16.9	861.8 ± 45.0	**980.8** ± 4.0
quadruped	walk	209.4 ± 55.9	820.5 ± 53.7	890.3 ± 47.4	778.1 ± 135.0	614.2 ± 234.7	817.7 ± 98.6	**948.3** ± 10.5
cartpole	swingup_sparse	615.1 ± 188.8	729.4 ± 148.4	800.5 ± 32.9	661.6 ± 196.7	824.0 ± 14.1	575.9 ± 221.7	**835.8** ± 8.7
reacher	hard	675.8 ± 115.8	530.3 ± 237.7	322.6 ± 188.7	436.5 ± 234.7	**874.2** ± 55.2	778.5 ± 200.7	651.3 ± 160.1
finger	turn_easy	231.1 ± 44.4	759.3 ± 81.1	660.5 ± 140.5	806.7 ± 58.3	853.2 ± 124.2	772.3 ± 120.8	**914.2** ± 18.7
walker	run	219.5 ± 76.2	464.8 ± 28.6	730.4 ± 12.1	747.0 ± 8.0	668.0 ± 12.9	368.3 ± 123.1	**794.9** ± 7.4
cheetah	run	134.3 ± 107.7	462.9 ± 61.3	529.4 ± 102.4	668.4 ± 11.9	673.8 ± 23.0	667.2 ± 47.0	**858.6** ± 16.4
pendulum	swingup	389.6 ± 220.7	774.3 ± 128.6	165.3 ± 204.8	836.4 ± 5.4	380.8 ± 255.2	364.2 ± 265.2	**841.2** ± 4.9
quadruped	run	176.5 ± 41.0	446.3 ± 20.4	542.9 ± 39.3	462.3 ± 55.0	360.3 ± 71.1	435.6 ± 21.6	**868.4** ± 29.1
finger	turn_hard	102.2 ± 20.6	588.7 ± 158.8	242.6 ± 152.8	580.2 ± 133.3	774.9 ± 192.2	846.1 ± 35.6	**880.9** ± 25.8
hopper	stand	11.0 ± 9.4	255.0 ± 150.3	36.8 ± 14.0	411.6 ± 160.4	219.7 ± 61.4	517.2 ± 193.2	**895.5** ± 24.8
acrobot	swingup	33.2 ± 11.6	**257.2** ± 28.4	9.5 ± 2.6	86.3 ± 42.5	68.6 ± 69.0	10.9 ± 2.2	214.2 ± 53.9
hopper	hop	0.8 ± 0.2	5.3 ± 6.5	5.0 ± 2.5	107.2 ± 36.8	10.1 ± 9.0	144.6 ± 32.7	**252.4** ± 8.0

Table 9: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=32.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
ball_in_cup	catch	881.7 ± 157.0	969.1 ± 3.7	940.7 ± 56.6	974.0 ± 1.7	972.5 ± 2.2	972.7 ± 1.5	**978.5** ± 0.8
cartpole	balance	863.2 ± 31.4	972.7 ± 10.2	616.4 ± 262.2	982.3 ± 14.5	964.7 ± 22.2	948.6 ± 21.6	**999.2** ± 0.8
cartpole	balance_sparse	843.9 ± 93.4	953.7 ± 27.6	703.9 ± 136.3	908.6 ± 68.6	919.7 ± 59.9	934.1 ± 45.5	**990.7** ± 11.7
walker	stand	527.4 ± 166.8	956.6 ± 9.9	827.2 ± 114.0	974.6 ± 3.9	968.7 ± 5.1	781.8 ± 122.4	**977.1** ± 3.6
cartpole	swingup	824.7 ± 22.7	770.3 ± 100.2	829.9 ± 78.8	741.2 ± 99.6	**865.7** ± 1.9	837.1 ± 9.1	865.3 ± 6.4
walker	walk	561.2 ± 116.3	852.9 ± 50.4	895.6 ± 57.1	959.5 ± 8.2	949.3 ± 4.9	805.0 ± 44.4	**959.6** ± 13.7
reacher	easy	487.0 ± 143.6	650.4 ± 169.1	875.3 ± 57.9	547.4 ± 141.8	900.6 ± 79.7	900.0 ± 38.4	**959.5** ± 9.3
finger	spin	626.2 ± 202.6	826.6 ± 23.4	519.5 ± 170.5	904.8 ± 32.4	897.9 ± 22.1	851.5 ± 52.3	**981.4** ± 3.6
quadruped	walk	314.9 ± 151.6	753.4 ± 53.4	875.7 ± 61.5	922.6 ± 51.8	534.5 ± 201.9	752.1 ± 157.3	**948.0** ± 10.0
cartpole	swingup_sparse	703.6 ± 93.4	817.2 ± 8.6	566.6 ± 102.2	665.9 ± 121.6	753.3 ± 98.2	796.2 ± 11.9	**836.6** ± 9.4
reacher	hard	600.7 ± 118.0	607.6 ± 156.4	438.2 ± 135.5	640.7 ± 162.9	825.3 ± 108.1	**882.5** ± 71.6	864.5 ± 131.0
finger	turn_easy	201.1 ± 42.4	722.5 ± 94.8	404.3 ± 162.9	723.1 ± 104.7	830.1 ± 137.2	773.7 ± 84.9	**928.4** ± 17.6
walker	run	216.8 ± 69.5	432.9 ± 63.0	711.1 ± 15.0	725.3 ± 35.3	670.2 ± 16.5	416.5 ± 125.5	**774.2** ± 12.9
cheetah	run	21.3 ± 45.0	509.8 ± 21.5	435.9 ± 129.0	653.0 ± 22.0	646.1 ± 15.5	707.5 ± 23.8	**863.2** ± 21.1
pendulum	swingup	353.2 ± 257.3	827.5 ± 12.3	461.3 ± 243.1	770.3 ± 151.4	220.6 ± 238.5	284.9 ± 228.7	**833.8** ± 12.6
quadruped	run	267.3 ± 68.3	437.1 ± 18.7	492.2 ± 42.7	471.4 ± 13.8	416.1 ± 81.2	447.8 ± 16.5	**846.3** ± 41.6
finger	turn_hard	105.1 ± 17.2	603.1 ± 112.9	174.1 ± 61.9	728.5 ± 70.8	588.6 ± 178.1	513.3 ± 131.8	**908.7** ± 13.9
hopper	stand	33.8 ± 19.6	148.7 ± 76.6	34.7 ± 9.4	293.7 ± 152.9	109.7 ± 43.2	387.7 ± 196.9	**848.0** ± 112.5
acrobot	swingup	37.9 ± 10.0	137.2 ± 40.2	55.8 ± 35.9	76.5 ± 44.0	16.2 ± 3.7	12.5 ± 2.2	**263.5** ± 51.9
hopper	hop	2.8 ± 2.3	1.1 ± 0.3	6.4 ± 1.7	84.5 ± 18.7	9.4 ± 8.4	134.5 ± 26.5	**245.8** ± 13.3

Table 10: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=2, noise dim=32.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
ball_in_cup	catch	938.0 ± 16.3	968.9 ± 3.0	908.7 ± 78.4	974.9 ± 1.4	973.9 ± 1.2	972.0 ± 2.0	**978.2** ± 0.6
cartpole	balance	892.5 ± 44.6	956.9 ± 28.5	768.2 ± 242.0	982.4 ± 11.8	962.1 ± 21.2	948.8 ± 25.3	**997.8** ± 4.3
cartpole	balance_sparse	684.2 ± 234.4	956.2 ± 48.1	755.7 ± 159.8	962.7 ± 45.4	973.1 ± 31.2	973.6 ± 24.0	**990.5** ± 17.5
walker	stand	416.2 ± 166.5	930.5 ± 32.5	853.4 ± 89.4	969.5 ± 15.1	965.2 ± 9.5	763.8 ± 116.7	**977.9** ± 7.2
cartpole	swingup	711.8 ± 144.2	852.4 ± 7.9	864.4 ± 9.6	523.6 ± 199.3	854.6 ± 6.6	829.4 ± 11.2	**870.3** ± 8.6
walker	walk	501.3 ± 105.7	858.9 ± 50.4	922.2 ± 140.2	947.2 ± 15.2	933.7 ± 17.9	742.6 ± 134.5	**958.3** ± 10.3
reacher	easy	463.1 ± 227.2	613.7 ± 277.7	841.3 ± 151.1	509.1 ± 249.7	936.3 ± 28.8	948.6 ± 14.1	**957.7** ± 22.8
finger	spin	521.1 ± 214.8	811.3 ± 15.6	512.7 ± 134.7	902.1 ± 26.2	850.5 ± 13.1	873.2 ± 36.1	**973.3** ± 13.5
quadruped	walk	320.5 ± 98.4	780.4 ± 39.8	804.2 ± 208.6	808.8 ± 172.7	626.0 ± 213.3	714.2 ± 148.5	**954.2** ± 3.9
cartpole	swingup_sparse	665.0 ± 100.0	786.7 ± 17.7	514.1 ± 149.1	780.9 ± 14.8	741.8 ± 148.9	706.9 ± 146.2	**840.9** ± 6.4
reacher	hard	690.7 ± 144.1	658.9 ± 219.9	358.0 ± 211.9	648.8 ± 207.0	824.6 ± 167.1	807.9 ± 187.3	**923.8** ± 32.9
finger	turn_easy	230.0 ± 56.4	697.0 ± 99.8	273.4 ± 56.4	727.2 ± 77.3	712.2 ± 143.6	657.5 ± 125.2	**935.0** ± 14.5
walker	run	80.3 ± 52.7	435.4 ± 56.5	696.2 ± 13.9	731.7 ± 16.4	666.0 ± 11.5	410.2 ± 83.1	**777.6** ± 10.4
cheetah	run	162.9 ± 100.0	483.2 ± 16.7	232.4 ± 181.0	615.7 ± 17.6	615.9 ± 26.9	665.7 ± 60.9	**851.5** ± 15.2
pendulum	swingup	256.6 ± 233.7	820.5 ± 14.3	557.6 ± 202.7	732.6 ± 167.0	199.7 ± 222.2	382.9 ± 254.0	**835.1** ± 9.6
quadruped	run	166.9 ± 42.6	440.8 ± 21.8	552.7 ± 52.1	439.6 ± 51.6	379.0 ± 77.1	414.8 ± 48.6	**864.5** ± 40.2
finger	turn_hard	91.6 ± 18.0	499.7 ± 157.5	228.8 ± 134.7	590.9 ± 87.0	427.8 ± 195.8	399.3 ± 146.5	**897.2** ± 19.5
hopper	stand	17.8 ± 12.9	77.0 ± 64.7	32.2 ± 12.1	346.3 ± 185.8	99.8 ± 63.1	544.8 ± 178.8	**904.7** ± 20.4
acrobot	swingup	23.5 ± 11.3	75.4 ± 30.1	36.1 ± 24.8	102.6 ± 46.4	18.8 ± 18.7	10.9 ± 3.0	**244.1** ± 47.0
hopper	hop	0.4 ± 0.3	2.8 ± 2.8	4.3 ± 4.1	79.1 ± 26.9	0.2 ± 0.2	129.9 ± 41.5	**242.6** ± 15.6

Table 11: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=4, noise dim=32.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
ball_in_cup	catch	918.6 ± 41.8	967.0 ± 3.5	723.9 ± 139.3	972.2 ± 4.3	973.9 ± 1.8	960.3 ± 15.7	**977.2** ± 1.1
cartpole	balance	815.1 ± 91.9	957.9 ± 13.2	901.4 ± 171.1	977.8 ± 19.4	969.2 ± 11.5	931.9 ± 29.9	**996.4** ± 5.7
cartpole	balance_sparse	831.1 ± 158.1	917.9 ± 50.4	830.1 ± 131.0	932.6 ± 103.1	**996.2** ± 3.6	899.8 ± 86.1	991.1 ± 17.2
walker	stand	394.7 ± 180.1	929.3 ± 28.9	875.1 ± 57.7	971.1 ± 6.4	951.9 ± 19.4	641.9 ± 146.9	**978.8** ± 5.8
cartpole	swingup	770.4 ± 34.2	850.4 ± 5.4	856.4 ± 21.9	684.2 ± 176.0	858.5 ± 5.6	823.9 ± 15.7	**881.2** ± 0.5
walker	walk	323.7 ± 131.3	834.2 ± 56.4	925.5 ± 30.6	954.7 ± 5.1	925.2 ± 26.5	778.4 ± 119.8	**965.8** ± 4.5
reacher	easy	312.9 ± 184.3	732.4 ± 244.1	616.1 ± 234.2	579.6 ± 269.9	950.6 ± 23.8	828.8 ± 110.3	**951.3** ± 18.1
finger	spin	220.8 ± 190.4	719.2 ± 68.4	528.9 ± 134.6	**917.6** ± 22.1	681.0 ± 144.3	862.5 ± 50.0	875.1 ± 46.6
quadruped	walk	204.0 ± 65.6	700.7 ± 105.9	768.2 ± 206.0	709.7 ± 147.2	506.2 ± 209.1	878.9 ± 27.7	**953.6** ± 9.1
cartpole	swingup_sparse	468.4 ± 219.8	562.2 ± 180.6	459.9 ± 198.6	617.6 ± 182.8	65.2 ± 143.5	385.4 ± 256.0	**841.4** ± 9.2
reacher	hard	369.3 ± 239.1	615.5 ± 221.1	389.7 ± 203.8	624.0 ± 230.1	580.1 ± 280.0	923.0 ± 32.1	**942.7** ± 12.6
finger	turn_easy	184.8 ± 28.4	503.9 ± 88.5	229.7 ± 41.4	724.5 ± 57.0	584.0 ± 171.8	491.6 ± 80.8	**927.2** ± 10.9
walker	run	57.6 ± 46.2	462.2 ± 43.1	674.3 ± 15.2	730.9 ± 19.1	646.4 ± 20.1	432.1 ± 118.4	**766.5** ± 13.0
cheetah	run	185.9 ± 79.0	459.6 ± 24.6	386.2 ± 154.2	575.5 ± 25.8	607.8 ± 13.2	657.4 ± 41.2	**857.6** ± 20.8
pendulum	swingup	123.2 ± 159.2	831.8 ± 9.2	577.4 ± 196.0	777.4 ± 130.4	72.3 ± 151.5	162.9 ± 194.8	**838.7** ± 4.0
quadruped	run	208.0 ± 48.5	452.7 ± 16.6	495.4 ± 92.6	476.3 ± 46.3	322.6 ± 86.0	434.1 ± 24.7	**850.7** ± 25.8
finger	turn_hard	100.5 ± 19.7	312.6 ± 92.0	135.7 ± 45.0	537.5 ± 117.1	371.1 ± 152.8	215.3 ± 83.6	**904.8** ± 20.6
hopper	stand	5.3 ± 0.6	30.3 ± 25.9	25.2 ± 11.0	179.1 ± 101.0	21.9 ± 25.2	332.9 ± 200.1	**828.0** ± 114.3
acrobot	swingup	19.4 ± 10.4	43.2 ± 28.5	16.6 ± 13.8	78.4 ± 45.2	11.4 ± 2.7	10.7 ± 2.1	**267.3** ± 49.5
hopper	hop	0.2 ± 0.2	0.7 ± 0.3	1.3 ± 1.0	75.0 ± 22.7	0.1 ± 0.0	51.8 ± 28.0	**255.7** ± 13.1

Table 12: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=8, noise dim=32.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
ball_in_cup	catch	924.7 ± 20.2	965.2 ± 3.9	760.4 ± 142.3	969.3 ± 4.8	924.8 ± 95.5	957.7 ± 22.7	**977.0** ± 1.0
cartpole	balance	814.1 ± 86.6	966.6 ± 9.2	950.3 ± 71.2	973.7 ± 12.4	967.5 ± 12.3	928.7 ± 32.3	**999.5** ± 0.5
cartpole	balance_sparse	646.6 ± 200.8	935.6 ± 34.8	784.0 ± 139.7	982.3 ± 29.3	984.7 ± 11.4	823.5 ± 160.2	**985.0** ± 15.4
walker	stand	265.4 ± 118.4	932.2 ± 13.8	796.8 ± 63.3	975.8 ± 5.1	968.6 ± 8.1	522.5 ± 154.5	**978.5** ± 5.5
cartpole	swingup	778.1 ± 28.9	851.9 ± 8.6	865.1 ± 21.4	838.7 ± 9.4	858.1 ± 5.7	828.6 ± 11.5	**878.8** ± 4.9
walker	walk	115.8 ± 60.1	834.5 ± 78.2	817.5 ± 111.6	954.6 ± 8.8	919.9 ± 23.6	537.1 ± 133.7	**966.7** ± 3.0
reacher	easy	306.0 ± 135.1	585.7 ± 274.6	552.9 ± 232.2	605.0 ± 201.1	896.6 ± 70.9	906.6 ± 22.3	**945.9** ± 44.7
finger	spin	174.4 ± 171.0	560.1 ± 50.3	408.0 ± 54.1	841.9 ± 88.1	226.0 ± 155.0	767.5 ± 115.8	**849.3** ± 53.8
quadruped	walk	255.5 ± 75.2	777.8 ± 61.2	805.9 ± 134.0	804.2 ± 113.3	407.6 ± 237.5	690.5 ± 142.0	**953.0** ± 4.8
cartpole	swingup_sparse	526.6 ± 169.5	526.0 ± 202.3	199.2 ± 150.1	613.2 ± 181.5	0.0 ± 0.0	318.3 ± 250.6	**844.0** ± 1.8
reacher	hard	395.4 ± 245.3	367.7 ± 231.4	157.7 ± 100.0	578.9 ± 163.3	630.9 ± 247.0	808.4 ± 173.4	**950.8** ± 8.1
finger	turn_easy	201.9 ± 38.5	419.0 ± 75.9	240.6 ± 36.4	619.0 ± 35.1	592.9 ± 176.6	327.3 ± 88.5	**926.8** ± 10.9
walker	run	23.9 ± 2.6	455.9 ± 41.3	649.4 ± 11.1	628.9 ± 25.7	635.3 ± 19.8	347.8 ± 84.0	**760.6** ± 19.4
cheetah	run	185.6 ± 86.7	433.8 ± 34.2	335.0 ± 87.5	553.2 ± 26.4	578.8 ± 22.0	628.9 ± 49.6	**866.3** ± 10.1
pendulum	swingup	124.1 ± 155.1	753.1 ± 148.5	462.9 ± 222.3	736.8 ± 243.9	4.0 ± 3.2	10.9 ± 2.7	**840.6** ± 5.2
quadruped	run	219.5 ± 63.5	417.9 ± 44.2	441.1 ± 93.7	433.3 ± 47.3	233.8 ± 59.0	381.1 ± 64.9	**847.4** ± 21.7
finger	turn_hard	97.9 ± 11.8	207.2 ± 53.8	110.8 ± 17.0	414.7 ± 49.5	177.6 ± 66.1	168.3 ± 50.4	**885.4** ± 24.5
hopper	stand	5.8 ± 0.5	29.1 ± 21.1	25.1 ± 14.4	198.1 ± 103.0	11.4 ± 10.7	116.6 ± 45.1	**899.1** ± 18.4
acrobot	swingup	11.1 ± 5.8	19.8 ± 10.9	17.0 ± 9.8	72.5 ± 43.9	14.3 ± 5.6	9.8 ± 3.6	**280.8** ± 32.6
hopper	hop	0.3 ± 0.3	0.4 ± 0.3	0.8 ± 0.5	51.1 ± 13.4	0.1 ± 0.0	31.3 ± 16.7	**233.9** ± 22.6

Table 13: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=2.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
ball_in_cup	catch	935.1 ± 19.2	973.5 ± 2.1	973.2 ± 3.2	973.5 ± 2.7	975.4 ± 1.2	970.1 ± 5.3	**978.0** ± 1.0
cartpole	balance	894.1 ± 36.9	648.2 ± 248.1	934.6 ± 56.1	983.9 ± 22.2	950.5 ± 18.7	935.2 ± 27.7	**990.8** ± 18.6
cartpole	balance_sparse	890.2 ± 81.4	548.1 ± 204.1	885.9 ± 48.8	861.6 ± 124.5	932.6 ± 80.2	**976.8** ± 19.6	949.6 ± 85.3
walker	stand	737.8 ± 123.8	951.9 ± 9.4	907.9 ± 31.3	970.3 ± 8.3	957.6 ± 17.1	903.4 ± 38.6	**976.6** ± 9.3
cartpole	swingup	806.9 ± 42.5	835.2 ± 36.0	**874.8** ± 1.6	757.6 ± 118.0	862.9 ± 10.2	844.1 ± 14.8	873.9 ± 5.5
walker	walk	635.2 ± 123.5	847.0 ± 76.0	833.6 ± 135.6	961.6 ± 5.1	946.4 ± 4.3	821.9 ± 99.4	**964.7** ± 4.2
reacher	easy	637.4 ± 186.6	660.9 ± 262.6	714.6 ± 214.8	361.8 ± 201.3	858.6 ± 114.2	701.9 ± 212.0	**929.2** ± 18.6
finger	spin	789.3 ± 41.8	875.8 ± 28.6	440.7 ± 151.8	922.9 ± 33.5	969.0 ± 6.6	903.1 ± 29.2	**979.5** ± 7.1
quadruped	walk	270.3 ± 131.4	787.2 ± 56.5	851.2 ± 96.3	783.7 ± 150.7	760.2 ± 178.3	807.6 ± 129.3	**950.5** ± 11.0
cartpole	swingup_sparse	387.3 ± 250.8	813.4 ± 15.9	685.2 ± 100.0	718.1 ± 146.1	787.0 ± 87.2	643.8 ± 191.9	**834.8** ± 9.8
reacher	hard	430.6 ± 206.5	467.3 ± 225.3	185.3 ± 104.2	516.1 ± 249.5	784.3 ± 106.5	**837.9** ± 127.6	592.1 ± 165.2
finger	turn_easy	226.9 ± 46.8	705.8 ± 118.6	691.6 ± 185.4	753.7 ± 123.1	**934.9** ± 10.0	829.5 ± 98.3	912.3 ± 16.3
walker	run	245.5 ± 86.2	467.9 ± 23.5	733.7 ± 14.4	751.9 ± 10.0	681.1 ± 10.3	458.9 ± 84.5	**781.6** ± 13.9
cheetah	run	167.6 ± 98.0	493.7 ± 45.9	484.8 ± 106.5	662.6 ± 24.8	699.0 ± 13.2	688.6 ± 52.0	**851.4** ± 25.4
pendulum	swingup	513.3 ± 232.5	**840.2** ± 4.9	39.7 ± 55.0	767.2 ± 151.8	565.5 ± 259.6	412.3 ± 262.0	770.2 ± 152.5
quadruped	run	170.7 ± 52.5	448.3 ± 13.0	571.1 ± 60.6	472.0 ± 59.4	356.0 ± 90.1	396.8 ± 59.3	**882.9** ± 24.5
finger	turn_hard	131.0 ± 72.2	702.8 ± 118.4	329.1 ± 223.8	516.0 ± 175.0	689.5 ± 198.0	755.5 ± 113.1	**864.5** ± 47.5
hopper	stand	22.8 ± 18.7	199.8 ± 140.6	39.0 ± 13.0	363.7 ± 180.1	218.6 ± 59.1	351.6 ± 145.5	**901.3** ± 25.4
acrobot	swingup	25.6 ± 7.7	**280.6** ± 21.2	9.2 ± 3.0	83.0 ± 34.8	46.3 ± 41.4	11.2 ± 2.3	183.4 ± 66.7
hopper	hop	1.5 ± 2.0	2.4 ± 2.2	4.4 ± 2.3	83.2 ± 29.2	52.6 ± 39.9	110.0 ± 45.2	**250.5** ± 15.2

Table 14: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=16.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
ball_in_cup	catch	956.9 ± 4.9	971.9 ± 2.3	966.5 ± 10.6	975.5 ± 1.3	974.4 ± 1.8	966.5 ± 14.1	**978.1** ± 0.8
cartpole	balance	865.9 ± 56.5	932.1 ± 58.2	768.3 ± 210.6	990.9 ± 5.0	958.0 ± 31.8	933.1 ± 17.9	**996.4** ± 5.6
cartpole	balance_sparse	764.4 ± 120.6	820.0 ± 134.4	830.0 ± 79.4	727.8 ± 135.0	962.2 ± 31.3	937.8 ± 38.4	**987.1** ± 12.8
walker	stand	514.7 ± 122.9	940.2 ± 14.5	887.8 ± 58.6	976.5 ± 3.3	968.9 ± 6.8	819.0 ± 142.4	**981.3** ± 2.8
cartpole	swingup	756.9 ± 139.8	729.5 ± 185.0	869.9 ± 7.0	794.8 ± 113.9	865.3 ± 5.2	827.4 ± 20.2	**870.0** ± 7.1
walker	walk	586.1 ± 146.6	808.7 ± 67.5	887.8 ± 64.4	**959.2** ± 6.8	927.3 ± 32.6	822.6 ± 49.6	954.2 ± 10.4
reacher	easy	526.6 ± 224.1	766.1 ± 204.1	916.4 ± 45.5	600.0 ± 203.9	851.8 ± 160.0	942.1 ± 27.9	**962.1** ± 7.1
finger	spin	807.3 ± 67.2	798.6 ± 81.6	565.8 ± 169.4	919.1 ± 25.6	951.3 ± 14.6	837.0 ± 26.9	**982.9** ± 0.8
quadruped	walk	199.1 ± 52.4	790.1 ± 66.4	865.7 ± 93.9	762.7 ± 142.0	764.3 ± 125.4	883.0 ± 17.7	**943.3** ± 20.4
cartpole	swingup_sparse	617.0 ± 164.0	693.7 ± 182.5	728.9 ± 104.1	735.7 ± 139.5	829.8 ± 9.1	723.6 ± 145.9	**837.5** ± 9.3
reacher	hard	628.9 ± 169.2	574.4 ± 233.5	362.2 ± 162.2	375.9 ± 236.3	**870.5** ± 38.1	853.9 ± 170.0	817.5 ± 136.7
finger	turn_easy	204.0 ± 31.8	766.0 ± 86.4	381.7 ± 152.1	742.7 ± 131.9	847.8 ± 89.8	765.1 ± 92.6	**920.8** ± 23.9
walker	run	143.7 ± 54.3	450.5 ± 42.2	722.7 ± 14.8	743.5 ± 17.2	669.3 ± 17.8	446.7 ± 91.5	**795.2** ± 5.0
cheetah	run	69.3 ± 104.1	508.0 ± 30.6	522.6 ± 99.1	669.8 ± 13.4	662.9 ± 19.1	670.3 ± 64.1	**874.6** ± 14.5
pendulum	swingup	332.5 ± 251.6	**835.9** ± 5.6	291.1 ± 198.8	766.7 ± 150.9	372.0 ± 264.5	283.6 ± 238.8	765.3 ± 151.3
quadruped	run	193.9 ± 73.4	457.0 ± 11.7	537.0 ± 63.3	486.9 ± 29.8	421.8 ± 79.0	414.1 ± 59.0	**889.8** ± 41.5
finger	turn_hard	108.4 ± 16.3	702.2 ± 104.5	211.5 ± 143.4	637.6 ± 85.1	707.7 ± 159.4	649.8 ± 134.5	**879.2** ± 20.5
hopper	stand	97.2 ± 126.9	191.7 ± 128.6	41.4 ± 12.7	292.9 ± 123.9	209.6 ± 148.6	515.7 ± 183.8	**843.0** ± 82.2
acrobot	swingup	30.1 ± 12.6	222.6 ± 17.9	40.9 ± 27.3	59.8 ± 33.4	25.8 ± 13.7	12.6 ± 2.6	**245.9** ± 37.0
hopper	hop	3.7 ± 4.5	5.5 ± 5.5	6.8 ± 3.5	93.8 ± 35.2	18.4 ± 13.2	122.4 ± 37.4	**238.5** ± 28.4

Table 15: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=64.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
ball_in_cup	catch	810.6 ± 189.0	965.1 ± 2.7	903.0 ± 78.8	974.7 ± 1.4	853.7 ± 176.4	970.6 ± 3.1	**978.2** ± 1.0
cartpole	balance	889.0 ± 37.5	967.2 ± 11.1	506.0 ± 259.4	980.5 ± 16.9	978.8 ± 18.9	932.7 ± 29.9	**999.8** ± 0.1
cartpole	balance_sparse	885.8 ± 36.0	938.5 ± 35.3	741.4 ± 120.5	965.1 ± 37.6	875.1 ± 117.6	927.2 ± 79.9	**989.9** ± 13.1
walker	stand	384.4 ± 117.1	923.9 ± 29.8	776.7 ± 63.6	970.7 ± 9.7	965.9 ± 4.4	698.4 ± 163.0	**980.9** ± 3.3
cartpole	swingup	726.1 ± 142.4	853.0 ± 5.7	**874.1** ± 9.2	770.9 ± 108.4	855.8 ± 3.9	836.5 ± 12.7	870.7 ± 6.0
walker	walk	408.2 ± 89.8	828.3 ± 71.3	915.6 ± 42.6	944.7 ± 31.5	940.4 ± 7.5	802.7 ± 72.2	**964.8** ± 3.7
reacher	easy	538.6 ± 155.7	761.9 ± 211.0	778.9 ± 193.9	751.9 ± 148.1	950.7 ± 13.3	798.1 ± 209.1	**962.3** ± 9.2
finger	spin	460.5 ± 197.1	776.3 ± 45.8	496.9 ± 107.2	915.5 ± 28.1	837.1 ± 15.0	802.9 ± 54.3	**964.4** ± 17.7
quadruped	walk	233.6 ± 65.9	799.8 ± 52.6	824.5 ± 134.6	825.9 ± 77.3	795.8 ± 154.0	797.5 ± 117.1	**945.6** ± 12.2
cartpole	swingup_sparse	739.4 ± 67.3	793.7 ± 28.5	611.0 ± 162.1	777.3 ± 28.8	814.0 ± 20.4	798.8 ± 20.7	**838.2** ± 8.9
reacher	hard	493.5 ± 176.6	594.9 ± 266.4	508.8 ± 181.3	654.3 ± 209.1	904.2 ± 41.1	771.1 ± 197.9	**944.4** ± 14.2
finger	turn_easy	204.4 ± 21.4	708.6 ± 108.3	336.0 ± 116.2	718.1 ± 103.7	548.0 ± 204.3	500.6 ± 133.6	**929.3** ± 25.8
walker	run	138.3 ± 54.1	443.9 ± 43.2	706.2 ± 15.5	727.3 ± 22.7	646.4 ± 23.2	495.6 ± 82.7	**765.0** ± 15.7
cheetah	run	42.4 ± 63.3	487.7 ± 30.1	340.3 ± 121.6	616.1 ± 19.8	598.4 ± 19.9	696.8 ± 28.9	**841.1** ± 20.8
pendulum	swingup	203.6 ± 211.9	803.3 ± 43.7	455.6 ± 254.6	**839.1** ± 5.3	494.2 ± 269.3	497.8 ± 257.4	838.9 ± 6.6
quadruped	run	195.3 ± 69.0	438.4 ± 26.7	526.2 ± 72.2	456.8 ± 38.4	420.0 ± 80.5	419.9 ± 31.0	**863.6** ± 39.4
finger	turn_hard	102.0 ± 18.7	294.7 ± 87.1	174.0 ± 67.9	597.3 ± 58.8	279.0 ± 159.5	344.3 ± 142.0	**908.2** ± 16.3
hopper	stand	9.6 ± 8.7	35.3 ± 28.5	29.4 ± 10.5	351.2 ± 175.7	94.4 ± 53.1	580.2 ± 165.3	**903.7** ± 19.2
acrobot	swingup	17.3 ± 9.1	67.4 ± 22.3	29.5 ± 24.6	94.9 ± 55.1	13.9 ± 4.4	12.7 ± 3.5	**275.4** ± 32.1
hopper	hop	2.6 ± 4.3	1.4 ± 0.9	3.4 ± 2.4	92.9 ± 29.0	2.1 ± 3.8	120.8 ± 44.6	**245.9** ± 14.1

Table 16: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=128.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
ball_in_cup	catch	602.4 ± 260.0	964.0 ± 2.3	793.0 ± 143.8	969.4 ± 7.6	817.5 ± 216.7	966.6 ± 8.1	**973.7** ± 1.5
cartpole	balance	868.8 ± 55.3	818.9 ± 203.4	768.9 ± 222.2	982.9 ± 10.4	926.5 ± 47.6	943.7 ± 23.0	**998.9** ± 1.7
cartpole	balance_sparse	631.0 ± 158.5	972.9 ± 19.2	671.7 ± 147.7	956.1 ± 33.5	962.6 ± 46.1	865.8 ± 128.5	**990.4** ± 11.7
walker	stand	315.8 ± 91.8	925.4 ± 18.7	863.1 ± 72.6	972.1 ± 9.2	953.3 ± 18.0	765.7 ± 128.3	**980.8** ± 2.1
cartpole	swingup	779.2 ± 27.6	849.1 ± 8.6	864.6 ± 4.8	641.8 ± 194.5	854.4 ± 8.7	835.5 ± 8.2	**874.0** ± 5.0
walker	walk	239.1 ± 112.6	879.3 ± 46.0	899.3 ± 69.0	954.8 ± 6.0	910.8 ± 29.5	754.4 ± 158.0	**962.0** ± 6.1
reacher	easy	297.5 ± 157.4	621.3 ± 254.4	704.9 ± 288.4	799.6 ± 147.2	935.3 ± 31.1	859.9 ± 160.6	**954.5** ± 13.1
finger	spin	394.5 ± 194.0	666.3 ± 55.8	433.7 ± 78.0	852.5 ± 64.3	657.3 ± 53.4	842.4 ± 64.8	**934.7** ± 38.2
quadruped	walk	209.5 ± 49.9	789.1 ± 53.4	803.6 ± 117.2	708.7 ± 125.6	567.7 ± 222.2	822.5 ± 100.1	**944.7** ± 14.4
cartpole	swingup_sparse	665.0 ± 104.9	0.6 ± 0.7	212.2 ± 172.6	517.7 ± 225.3	397.2 ± 264.0	650.1 ± 194.3	**836.3**± 9.5
reacher	hard	712.8 ± 152.2	263.2 ± 213.3	242.2 ± 177.5	747.7 ± 158.7	693.0 ± 223.5	899.3 ± 85.7	**944.9** ± 10.6
finger	turn_easy	210.6 ± 46.1	477.5 ± 105.4	237.6 ± 62.4	655.3 ± 53.6	459.0 ± 118.5	374.4 ± 104.9	**936.8** ± 7.5
walker	run	26.2 ± 2.9	449.9 ± 62.1	703.9 ± 15.7	710.6 ± 28.9	643.3 ± 14.8	507.0 ± 97.3	**766.6** ± 12.6
cheetah	run	219.5 ± 80.4	441.5 ± 42.3	297.7 ± 142.4	508.5 ± 39.8	577.4 ± 23.6	648.1 ± 35.9	**862.4** ± 11.9
pendulum	swingup	85.8 ± 78.0	764.4 ± 122.2	741.3 ± 140.4	706.9 ± 176.5	349.4 ± 271.1	166.2 ± 197.5	**839.5** ± 5.6
quadruped	run	181.8 ± 52.9	446.5 ± 21.8	508.2 ± 61.6	408.6 ± 73.9	315.3 ± 72.3	409.5 ± 50.1	**867.3** ± 37.0
finger	turn_hard	94.3 ± 24.0	213.2 ± 60.2	126.2 ± 21.1	497.7 ± 49.4	278.4 ± 110.5	290.4 ± 140.9	**899.2** ± 27.3
hopper	stand	15.1 ± 12.4	28.7 ± 23.2	15.2 ± 8.7	261.3 ± 153.5	50.6 ± 35.8	524.3 ± 197.3	**874.0**± 42.3
acrobot	swingup	16.1 ± 8.7	24.9 ± 8.5	29.1 ± 14.5	101.9 ± 50.1	14.0 ± 3.0	10.5 ± 2.5	**243.2** ± 41.1
hopper	hop	0.4 ± 0.2	0.9 ± 0.1	1.8 ± 2.3	69.5 ± 26.3	0.1 ± 0.2	119.9 ± 28.8	**245.2** ± 16.9

Table 17: Performance of pixel-based DMC tasks for the compared methods with original clean background.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
cartpole	balance	963.7 ± 21.7	984.4 ± 9.1	981.9 ± 27.0	945.0 ± 27.8	**990.1** ± 5.1	943.2 ± 67.7	986.8 ± 5.8
cartpole	balance_sparse	617.1 ± 629.2	984.0 ± 29.1	992.7 ± 16.2	995.1 ± 6.1	**997.9** ± 3.9	916.3 ± 81.8	972.0 ± 47.9
walker	stand	923.8 ± 63.4	968.3 ± 7.7	**971.1** ± 6.3	963.2 ± 8.6	968.6 ± 9.0	961.1 ± 20.0	875.2 ± 219.7
finger	spin	837.5 ± 197.7	845.5 ± 231.1	967.1 ± 40.7	913.5 ± 179.2	918.2 ± 173.4	**973.9** ± 29.9	918.6 ± 181.1
cartpole	swingup	841.9 ± 35.7	856.1 ± 7.0	**859.5** ± 2.5	815.6 ± 21.8	852.3 ± 16.0	856.2 ± 15.6	838.6 ± 45.7
ball_in_cup	catch	311.0 ± 450.8	950.2 ± 20.4	**970.0** ± 1.9	843.5 ± 180.9	789.1 ± 481.4	793.9 ± 499.4	969.8 ± 6.8
walker	walk	658.8 ± 184.6	764.3 ± 18.9	936.6 ± 32.3	941.9 ± 25.7	785.8 ± 129.8	955.9 ± 16.0	**957.2** ± 4.1
point_mass	easy	**830.1** ± 69.4	512.5 ± 579.2	715.1 ± 495.6	418.7 ± 490.5	287.4 ± 506.9	490.4 ± 558.9	246.4 ± 483.4
cartpole	swingup_sparse	732.8 ± 18.8	780.7 ± 49.3	758.6 ± 42.0	745.0 ± 74.7	766.7 ± 45.5	**800.8** ± 30.6	655.2 ± 455.3
reacher	easy	148.5 ± 107.9	228.3 ± 50.2	**924.1** ± 44.5	212.7 ± 99.3	583.0 ± 147.5	129.9 ± 39.2	84.6 ± 19.7
pendulum	swingup	19.3 ± 4.7	814.8 ± 26.3	**841.6** ± 8.2	835.3 ± 11.0	424.5 ± 513.2	839.3 ± 6.7	668.9 ± 463.2
cheetah	run	223.9 ± 270.5	583.6 ± 98.3	**683.4** ± 221.2	302.2 ± 215.1	586.9 ± 154.5	403.7 ± 343.3	672.7 ± 465.3
walker	run	142.9 ± 111.1	480.6 ± 32.8	530.1 ± 225.6	**578.2** ± 47.8	304.3 ± 74.3	503.2 ± 171.3	550.5 ± 74.8
hopper	hop	18.8 ± 21.4	178.1 ± 21.2	108.5 ± 61.6	129.2 ± 90.0	160.9 ± 21.6	**203.1** ± 52.5	113.3 ± 136.8

Table 18: Performance of pixel-based DMC tasks for the compared methods in colored images background.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
cartpole	balance	977.8 ± 15.4	975.8 ± 14.2	932.5 ± 97.6	933.3 ± 28.1	979.3 ± 16.8	**982.1** ± 12.6	980.5 ± 16.8
cartpole	balance_sparse	781.5 ± 484.3	987.1 ± 25.3	**998.2** ± 3.5	825.9 ± 480.1	985.1 ± 14.6	998.2 ± 4.2	990.8 ± 18.9
walker	stand	883.3 ± 166.4	938.8 ± 57.4	**963.4** ± 8.1	958.7 ± 9.7	941.3 ± 53.9	949.7 ± 47.8	948.4 ± 26.4
finger	spin	489.8 ± 583.7	981.8 ± 2.9	786.8 ± 546.2	670.4 ± 504.0	975.3 ± 17.4	902.4 ± 207.2	**984.4** ± 4.0
cartpole	swingup	827.5 ± 25.7	827.5 ± 37.6	**856.6** ± 9.8	799.6 ± 11.9	851.0 ± 7.9	844.1 ± 32.5	707.1 ± 373.0
ball_in_cup	catch	119.9 ± 38.9	775.3 ± 454.1	**967.7** ± 6.3	613.6 ± 573.5	605.6 ± 599.0	918.8 ± 146.8	459.9 ± 579.6
walker	walk	635.0 ± 37.4	799.8 ± 162.9	907.9 ± 19.0	828.2 ± 125.8	701.7 ± 158.9	570.0 ± 620.9	**938.2** ± 23.8
point_mass	easy	258.2 ± 440.5	**710.3** ± 493.0	544.9 ± 602.0	527.0 ± 406.0	156.3 ± 424.4	524.1 ± 592.5	670.8 ± 468.6
cartpole	swingup_sparse	717.7 ± 49.0	787.5 ± 48.0	770.3 ± 59.1	551.3 ± 386.5	611.6 ± 429.4	**804.3** ± 51.0	797.2 ± 54.1
reacher	easy	129.0 ± 58.2	188.5 ± 46.0	**952.0** ± 7.6	164.6 ± 73.9	673.9 ± 125.1	195.8 ± 41.1	83.0 ± 20.9
pendulum	swingup	1.1 ± 0.7	**622.7** ± 650.5	414.1 ± 473.9	174.1 ± 466.3	70.4 ± 128.8	168.7 ± 452.5	336.1 ± 570.1
cheetah	run	212.8 ± 227.5	368.1 ± 31.9	373.6 ± 58.2	284.8 ± 196.5	362.2 ± 40.7	**377.3** ± 59.0	207.5 ± 376.6
walker	run	140.9 ± 96.7	385.2 ± 33.5	514.1 ± 68.3	531.3 ± nan	269.1 ± 45.2	368.6 ± 268.3	**543.1** ± 43.0
hopper	hop	9.2 ± 20.5	63.9 ± 77.2	71.2 ± 83.4	37.4 ± 117.6	110.7 ± 126.1	**174.0** ± 42.6	118.3 ± 133.4

Table 19: Performance of pixel-based DMC tasks for the compared methods in grayscale images background.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
cartpole	balance	788.0 ± 343.4	883.6 ± 221.3	986.4 ± 13.3	845.4 ± 262.1	**986.8** ± 6.3	943.3 ± 81.8	985.3 ± 6.9
cartpole	balance_sparse	791.7 ± 540.5	998.2 ± 2.3	997.6 ± 6.1	761.2 ± 751.8	999.4 ± 0.9	**999.4** ± 1.0	985.9 ± 16.6
walker	stand	822.1 ± 189.7	952.6 ± 13.5	**961.4** ± 10.0	922.8 ± 84.9	955.7 ± 23.0	950.5 ± 38.0	952.0 ± 20.2
finger	spin	558.5 ± 632.9	906.8 ± 210.6	955.1 ± 66.0	705.9 ± 499.9	972.2 ± 16.6	926.4 ± 155.9	**982.8** ± 4.8
cartpole	swingup	830.6 ± 24.5	839.9 ± 19.3	**861.5** ± 11.7	749.2 ± 115.1	855.9 ± 12.1	838.9 ± 21.3	507.9 ± 409.3
ball_in_cup	catch	205.1 ± 276.5	782.8 ± 396.7	**797.6** ± 484.3	568.6 ± 623.2	739.3 ± 483.9	761.1 ± 440.6	585.2 ± 580.9
walker	walk	502.5 ± 224.8	712.2 ± 137.8	922.1 ± 13.1	739.5 ± 495.3	640.0 ± 46.6	931.5 ± 66.0	**938.6** ± 26.4
point_mass	easy	518.8 ± 587.9	535.4 ± 604.8	**536.7** ± 605.3	201.0 ± 435.0	172.7 ± 477.7	324.6 ± 545.9	465.7 ± 414.0
cartpole	swingup_sparse	670.5 ± 65.7	**794.4** ± 20.3	354.8 ± 476.7	596.8 ± 358.2	610.0 ± 647.6	629.0 ± 437.1	785.0 ± 51.5
reacher	easy	141.6 ± 81.7	203.3 ± 70.9	**962.4** ± 19.4	179.4 ± 111.2	590.2 ± 152.4	150.7 ± 73.8	88.1 ± 23.6
pendulum	swingup	4.9 ± 5.7	**549.5** ± 572.5	199.7 ± 435.7	337.2 ± 553.5	59.3 ± 148.8	339.0 ± 569.5	346.2 ± 576.4
cheetah	run	202.0 ± 161.9	369.4 ± 56.0	**410.0** ± 29.6	217.3 ± 244.0	376.8 ± 31.9	286.5 ± 198.8	253.4 ± 285.4
walker	run	74.7 ± 209.4	416.2 ± 33.1	**469.4** ± 144.8	310.3 ± 327.2	281.3 ± 52.4	267.1 ± 313.3	404.5 ± 269.8
hopper	hop	11.5 ± 30.9	120.9 ± 88.7	79.9 ± 54.9	73.5 ± 82.4	**147.0** ± 19.8	100.2 ± 117.5	78.8 ± 134.1

Table 20: Performance of pixel-based DMC tasks for the compared methods in colored video background.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
cartpole	balance	924.7 ± 54.5	956.1 ± 45.7	961.8 ± 27.4	760.6 ± 386.2	979.4 ± 8.7	**985.6** ± 22.1	984.1 ± 8.9
cartpole	balance_sparse	677.1 ± 212.8	986.0 ± 22.6	958.2 ± 93.6	990.7 ± 14.0	986.8 ± 13.7	**997.5** ± 6.4	993.6 ± 20.2
walker	stand	534.5 ± 347.5	943.3 ± 17.7	960.0 ± 12.1	951.0 ± 5.3	**961.8** ± 10.8	916.2 ± 128.5	957.7 ± 5.6
finger	spin	163.9 ± 521.6	978.8 ± 12.7	860.1 ± 331.9	905.2 ± 177.5	962.1 ± 27.0	**982.2** ± 4.6	908.8 ± 212.0
cartpole	swingup	770.6 ± 40.8	850.5 ± 7.6	847.6 ± 33.3	301.0 ± 347.0	847.0 ± 11.9	**861.8** ± 33.1	832.9 ± 72.4
ball_in_cup	catch	95.7 ± 57.6	885.1 ± 119.2	800.5 ± 533.7	766.6 ± 412.8	**935.9** ± 55.0	365.6 ± 262.3	885.4 ± 94.9
walker	walk	100.6 ± 144.0	845.3 ± 83.9	**921.6** ± 30.3	651.9 ± 458.0	684.6 ± 151.0	667.6 ± 452.6	912.6 ± 26.2
point_mass	easy	2.2 ± 3.8	**889.8** ± 5.7	25.2 ± 31.6	281.4 ± 670.6	889.6 ± 10.1	365.6 ± 592.9	178.3 ± 477.2
cartpole	swingup_sparse	0.0 ± 0.0	731.2 ± 91.1	10.3 ± 14.5	92.1 ± 292.3	**734.0** ± 97.4	573.0 ± 429.4	675.7 ± 240.7
reacher	easy	203.8 ± 84.7	188.6 ± 26.2	**949.2** ± 45.9	175.2 ± 76.3	249.6 ± 66.1	183.3 ± 59.1	86.4 ± 18.0
pendulum	swingup	6.3 ± 8.1	**733.2** ± 351.4	452.2 ± 453.2	334.7 ± 549.6	85.3 ± 93.9	519.2 ± 419.7	654.8 ± 447.6
cheetah	run	53.0 ± 59.8	341.2 ± 34.0	**395.4** ± 21.2	220.3 ± 49.8	348.8 ± 8.4	306.7 ± 37.8	315.7 ± 21.0
walker	run	54.9 ± 23.3	367.0 ± 33.4	**492.7** ± 135.4	148.5 ± 153.7	244.1 ± 70.8	115.8 ± 103.8	168.8 ± 170.5
hopper	hop	0.2 ± 0.4	108.2 ± 80.7	41.2 ± 45.3	27.6 ± 46.0	**136.1** ± 18.9	23.0 ± 37.0	27.5 ± 50.0

Table 21: Performance of pixel-based DMC tasks for the compared methods in grayscale video background.

Task		DBC	MICo	RAP	RDBC	SAC	DeepMDP	SimSR
cartpole	balance	951.9 ± 36.1	977.2 ± 6.6	983.2 ± 12.6	882.2 ± 271.8	973.3 ± 17.5	**990.7** ± 6.7	962.9 ± 57.5
cartpole	balance_sparse	875.0 ± 134.2	996.3 ± 2.2	956.8 ± 67.6	803.4 ± 545.8	991.0 ± 15.2	**996.4** ± 8.9	990.4 ± 23.0
walker	stand	643.8 ± 253.2	947.8 ± 25.7	**971.0** ± 3.9	961.1 ± 3.1	961.5 ± 3.7	946.6 ± 64.9	957.0 ± 8.7
finger	spin	0.0 ± 0.0	979.9 ± 7.7	971.4 ± 21.4	974.8 ± 13.1	976.9 ± 11.8	**982.7** ± 2.9	902.6 ± 261.3
cartpole	swingup	849.1 ± 17.5	854.3 ± 11.1	861.7 ± 8.0	389.8 ± 388.4	858.0 ± 14.2	**863.4** ± 12.3	854.7 ± 51.0
ball_in_cup	catch	118.0 ± 39.2	930.5 ± 28.0	958.9 ± 23.3	865.0 ± 144.5	**962.0** ± 3.6	693.5 ± 259.7	907.4 ± 70.0
walker	walk	146.5 ± 84.5	763.3 ± 258.4	**913.6** ± 33.1	885.0 ± 46.9	650.4 ± 191.3	869.2 ± 25.5	907.8 ± 48.7
point_mass	easy	156.8 ± 428.7	**893.0** ± 11.0	439.9 ± 356.9	70.8 ± 195.6	885.1 ± 28.4	709.9 ± 492.9	541.6 ± 596.3
cartpole	swingup_sparse	0.0 ± 0.0	**803.0** ± 26.9	387.9 ± 440.7	339.2 ± 400.9	643.4 ± 447.7	643.8 ± 449.4	490.0 ± 555.5
reacher	easy	162.8 ± 115.3	171.1 ± 69.7	**968.2** ± 14.7	166.9 ± 42.4	268.3 ± 83.7	209.2 ± 53.8	84.9 ± 16.5
pendulum	swingup	33.9 ± 90.5	695.5 ± 391.8	708.2 ± 290.2	269.1 ± 596.6	207.5 ± 459.2	**836.2** ± 17.7	835.6 ± 15.4
cheetah	run	77.4 ± 62.4	366.9 ± 12.9	**404.8** ± 20.8	170.2 ± 194.0	359.9 ± 25.5	343.2 ± 69.7	343.4 ± 18.5
walker	run	63.1 ± 17.1	393.4 ± 16.2	**544.0** ± 26.3	180.3 ± 109.5	244.8 ± 45.7	175.8 ± 106.2	270.5 ± 190.6
hopper	hop	0.1 ± 0.1	117.3 ± 81.9	77.3 ± 53.2	38.2 ± 66.9	**121.1** ± 84.2	18.3 ± 49.4	0.9 ± 0.9

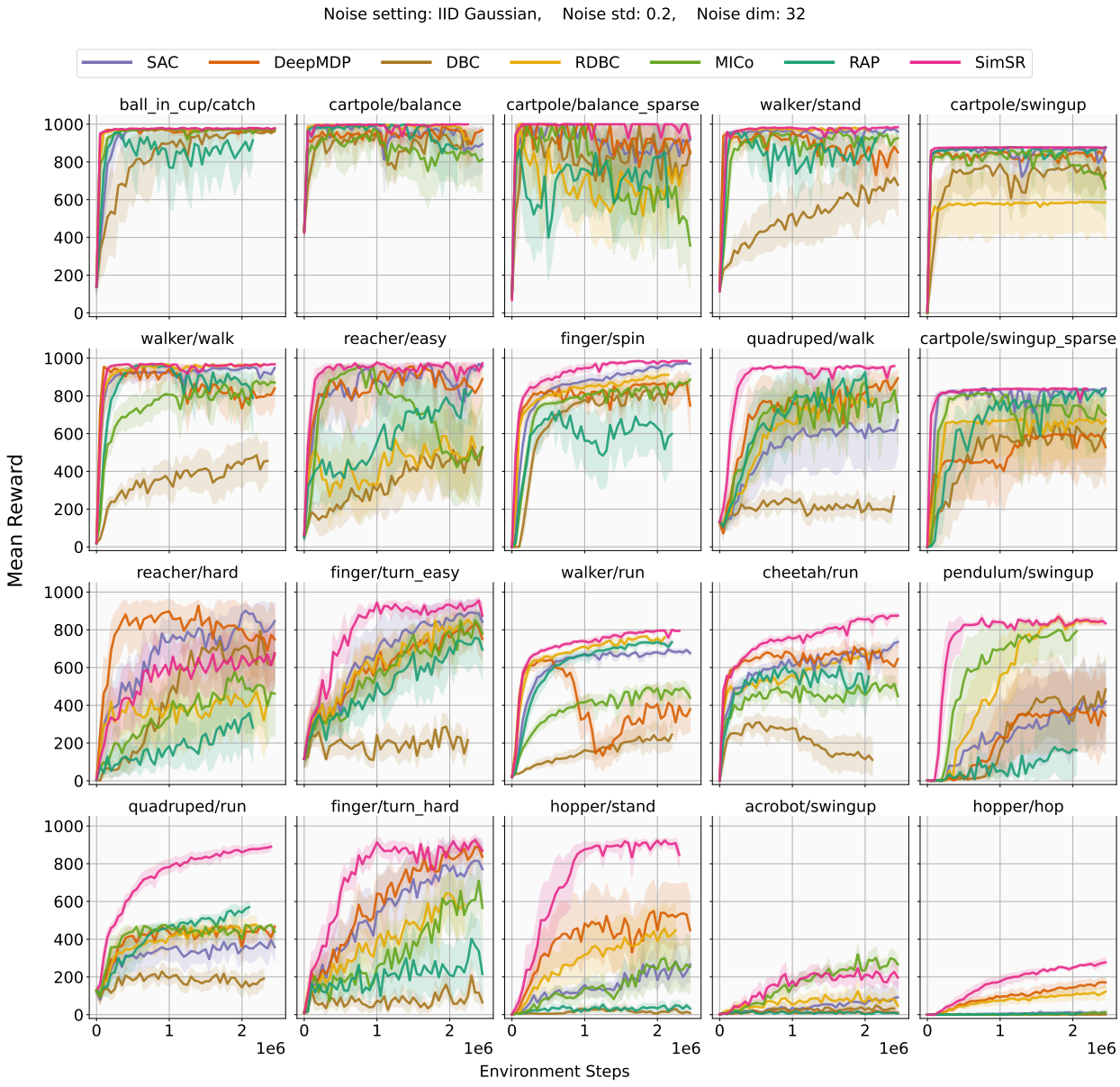

Figure 10: Individual state-based task performance.

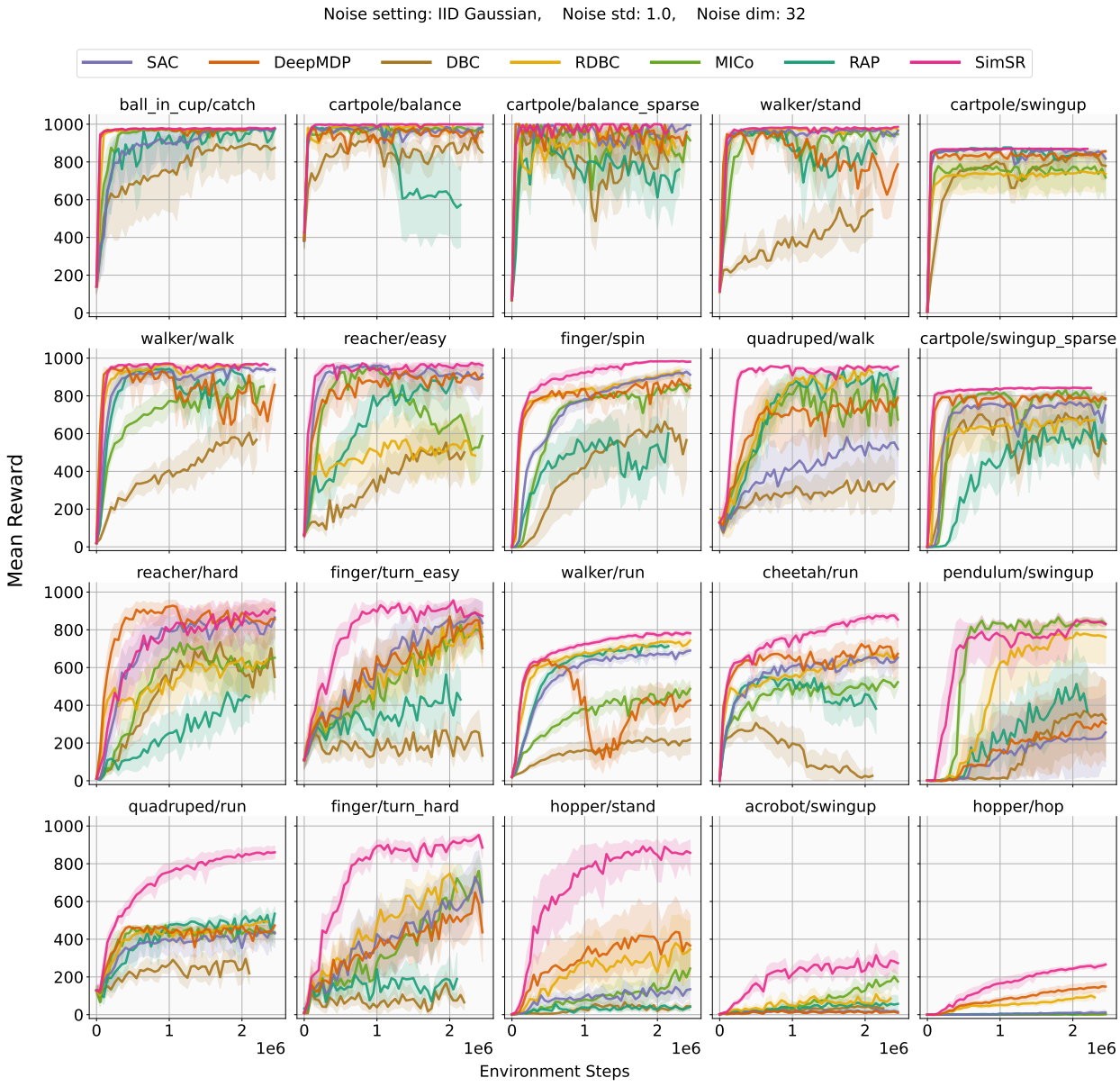

Figure 11: Individual state-based task performance.

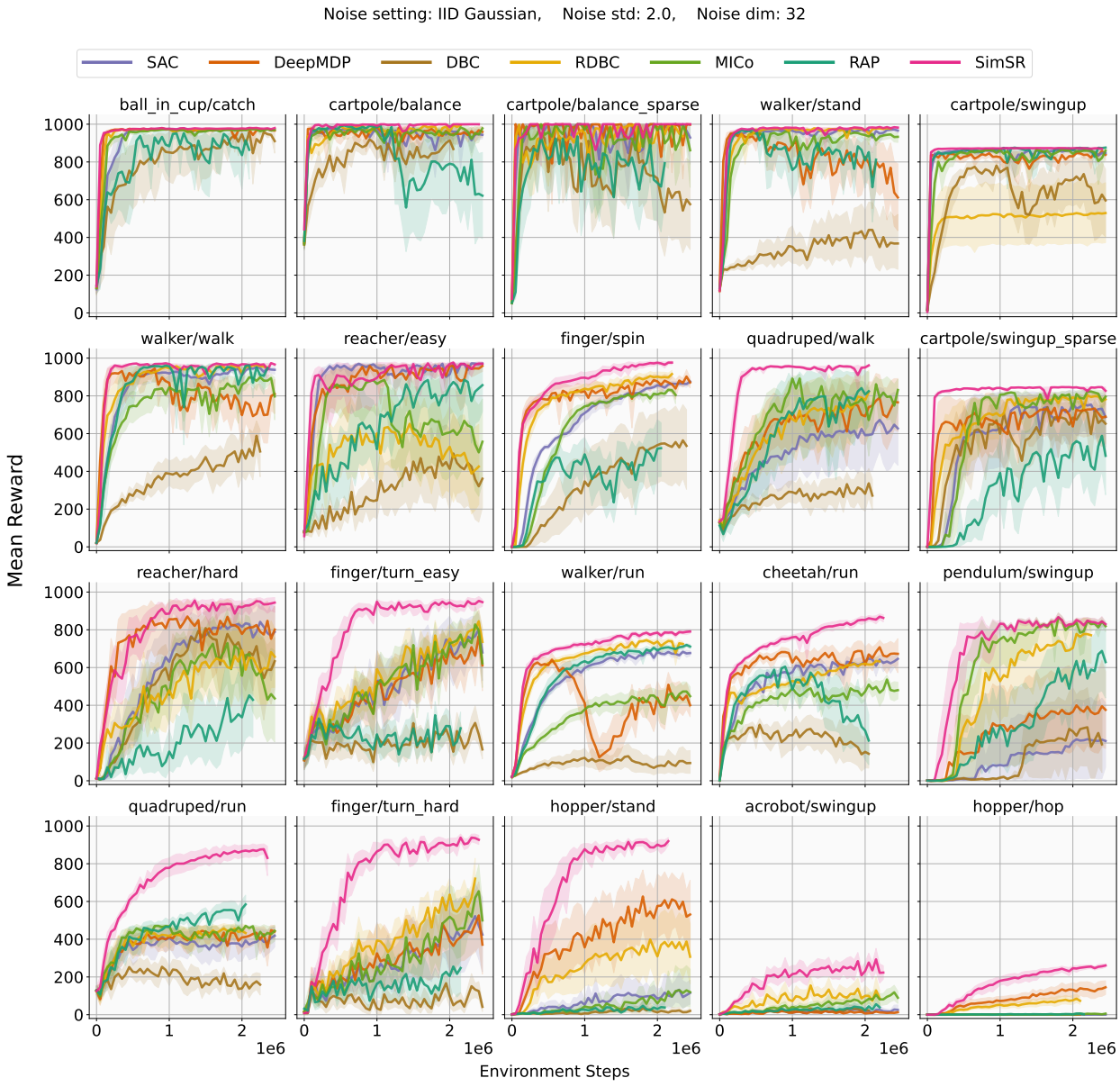

Figure 12: Individual state-based task performance.

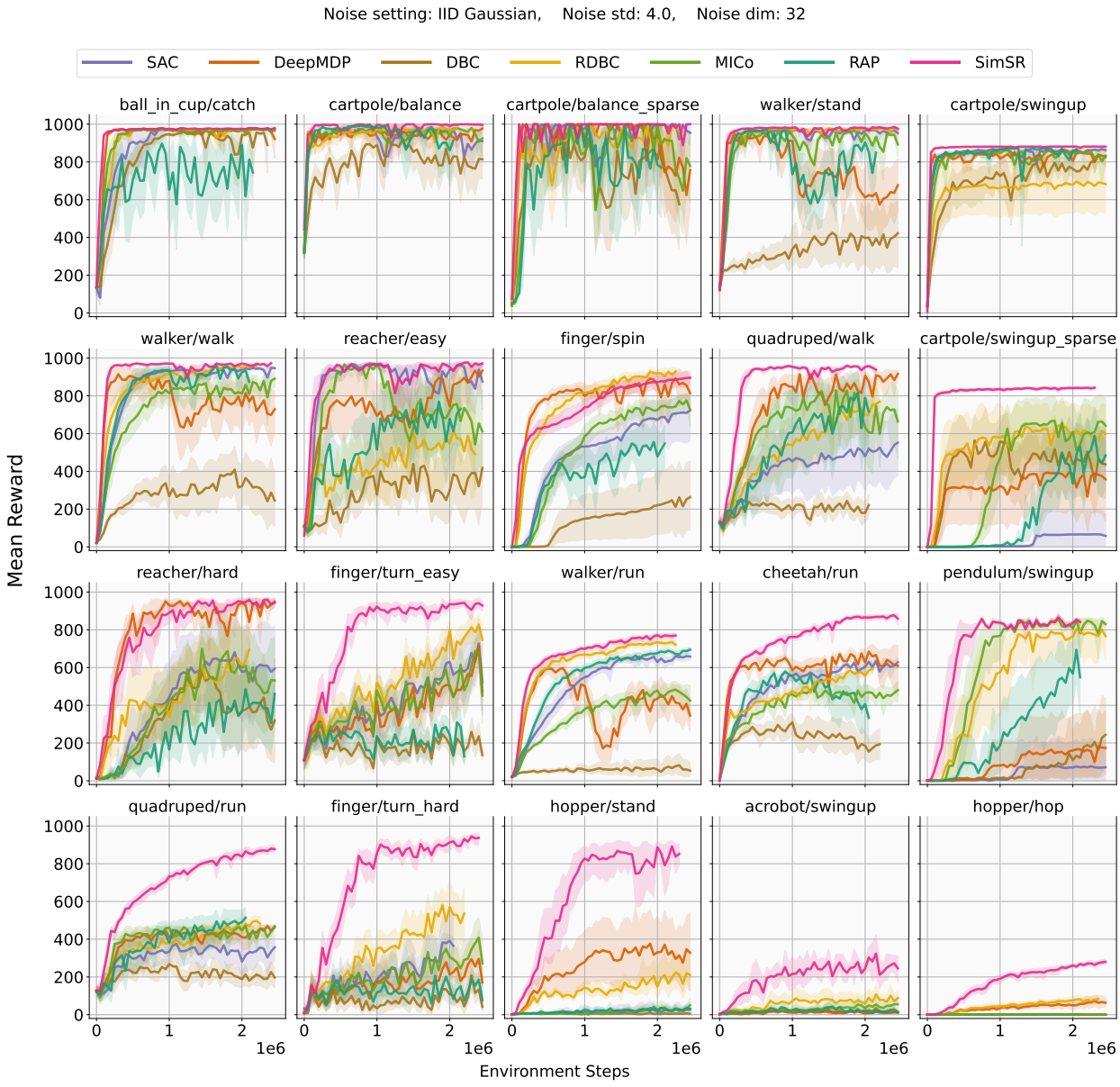

Figure 13: Individual state-based task performance.

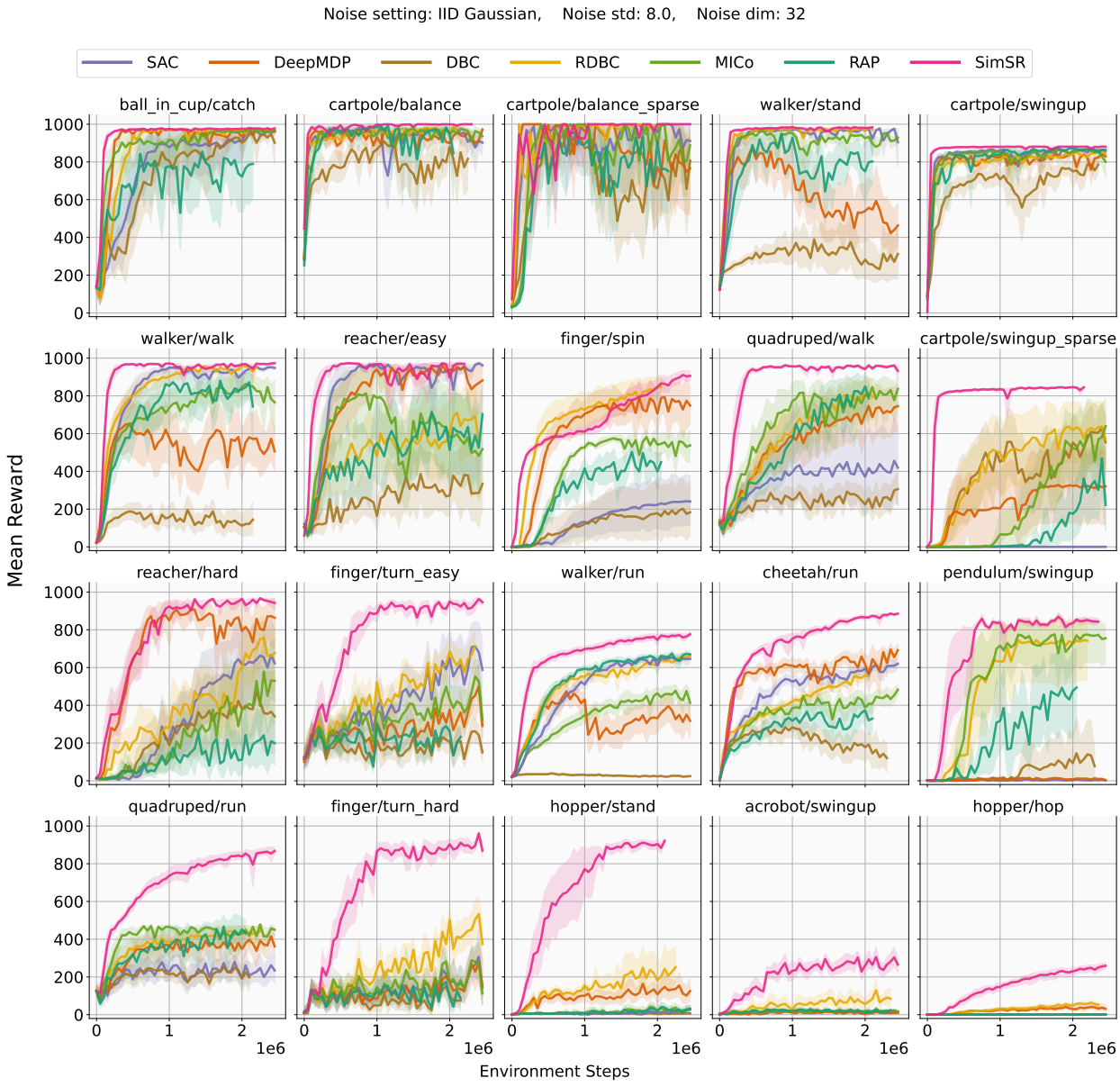

Figure 14: Individual state-based task performance.

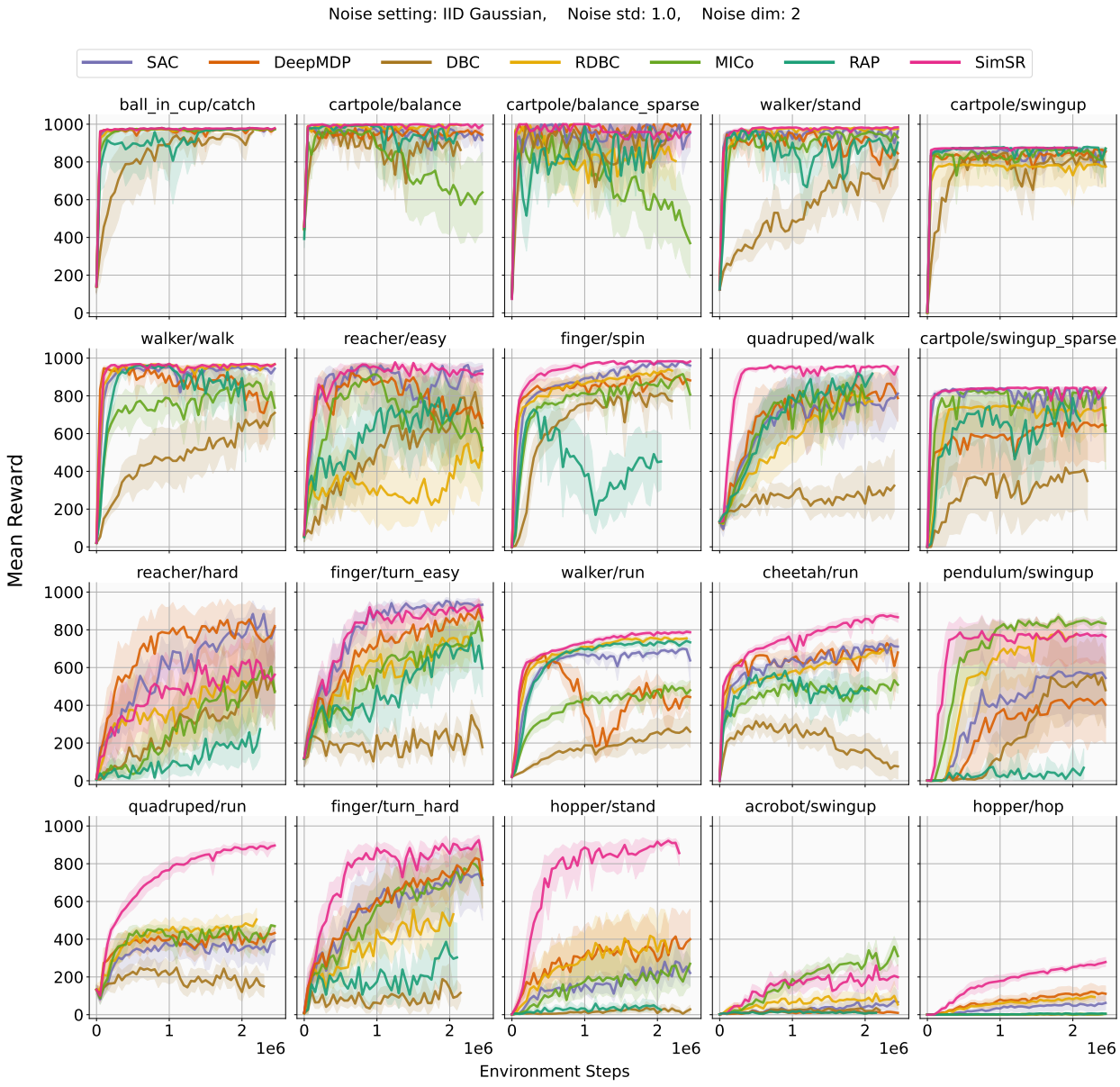

Figure 15: Individual state-based task performance.

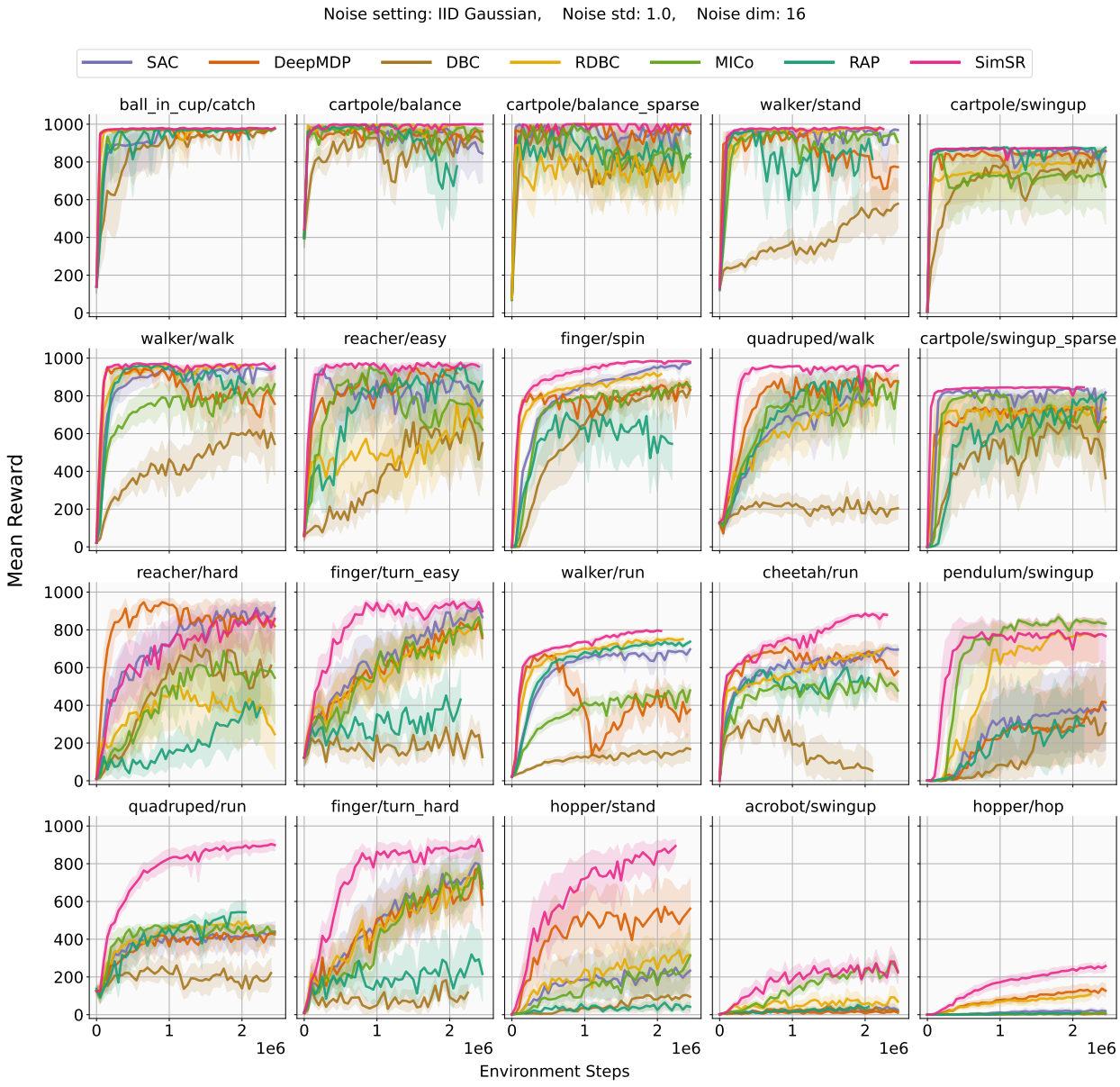

Figure 16: Individual state-based task performance.

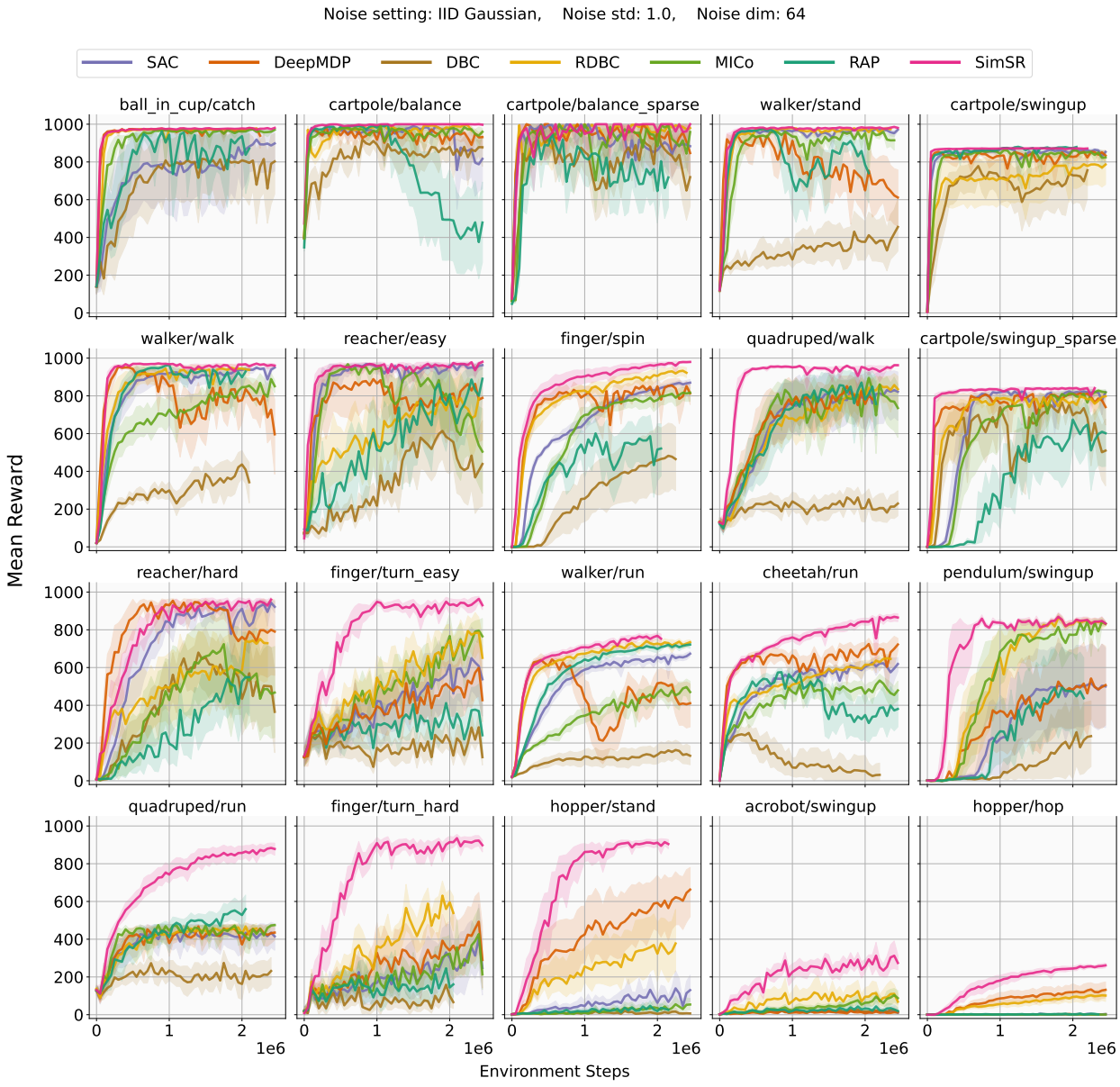

Figure 17: Individual state-based task performance.

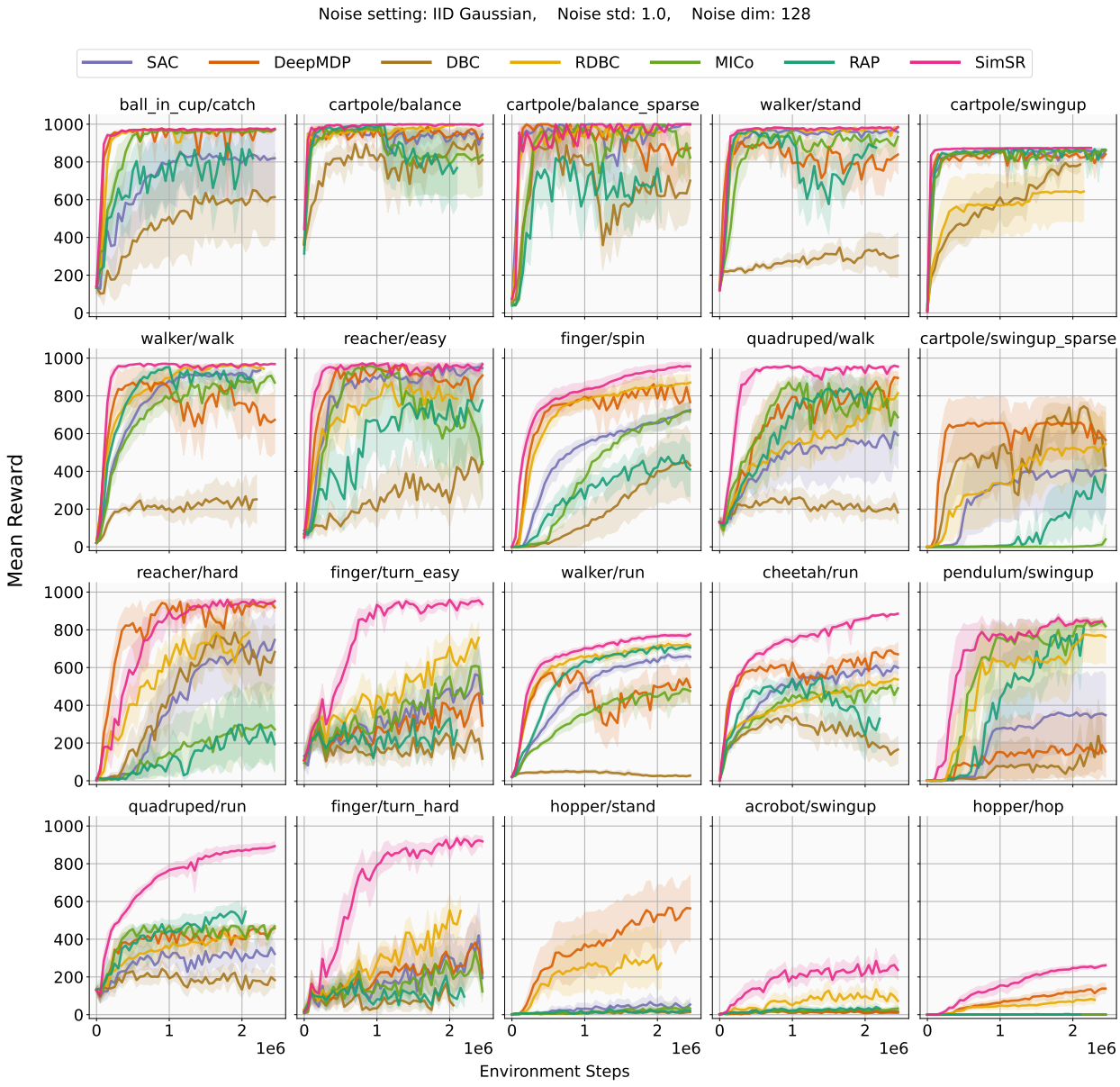

Figure 18: Individual state-based task performance.

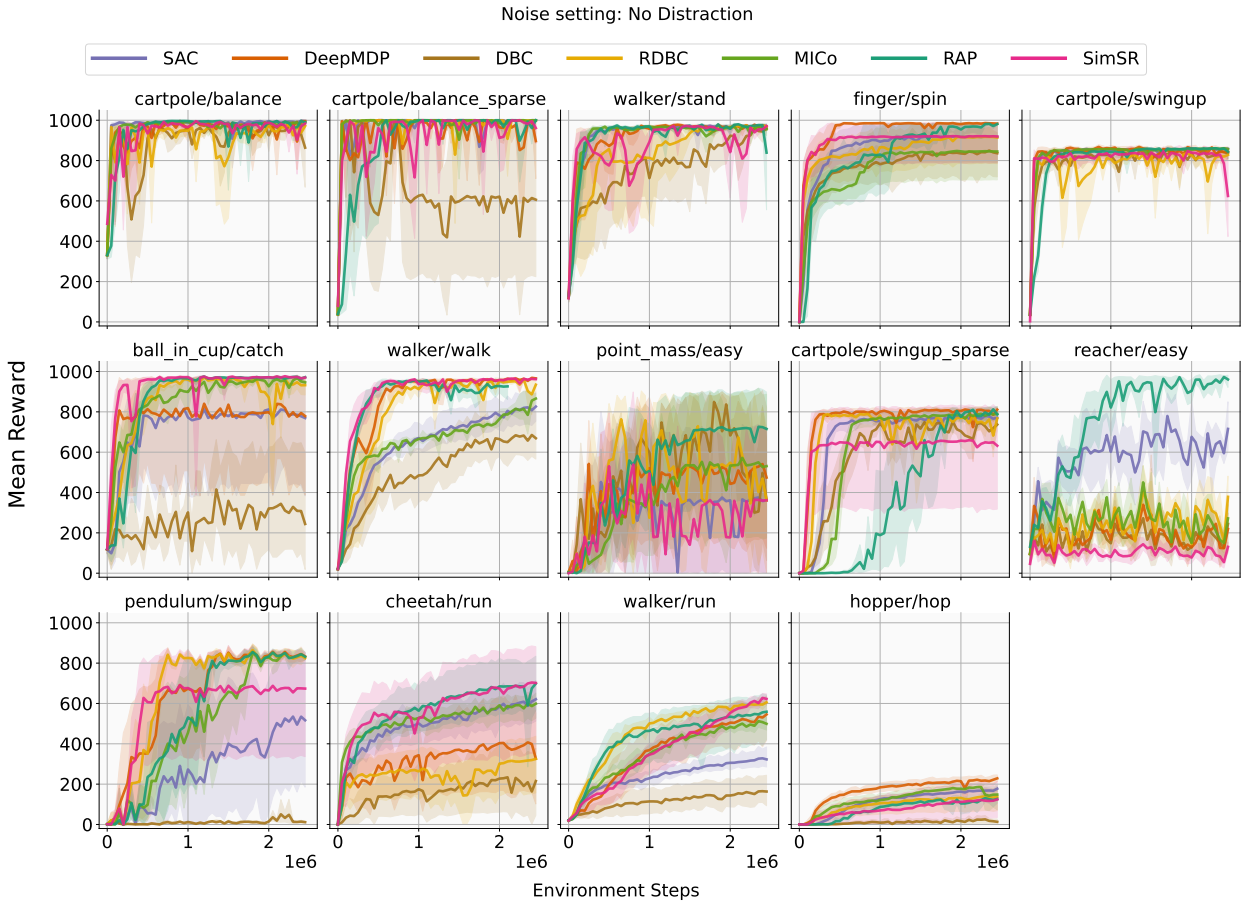

Figure 19: Individual pixel-based task performance.

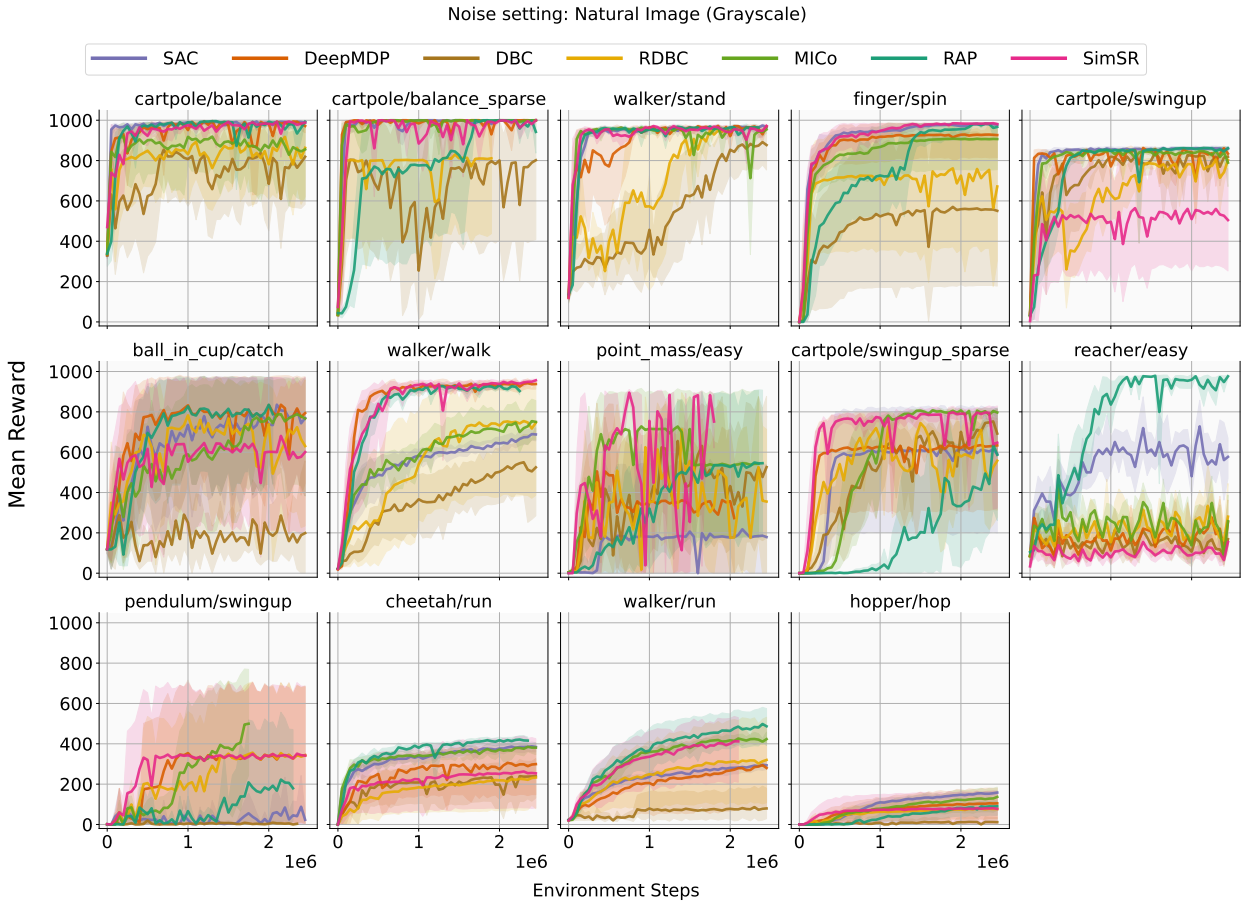

Figure 20: Individual pixel-based task performance.

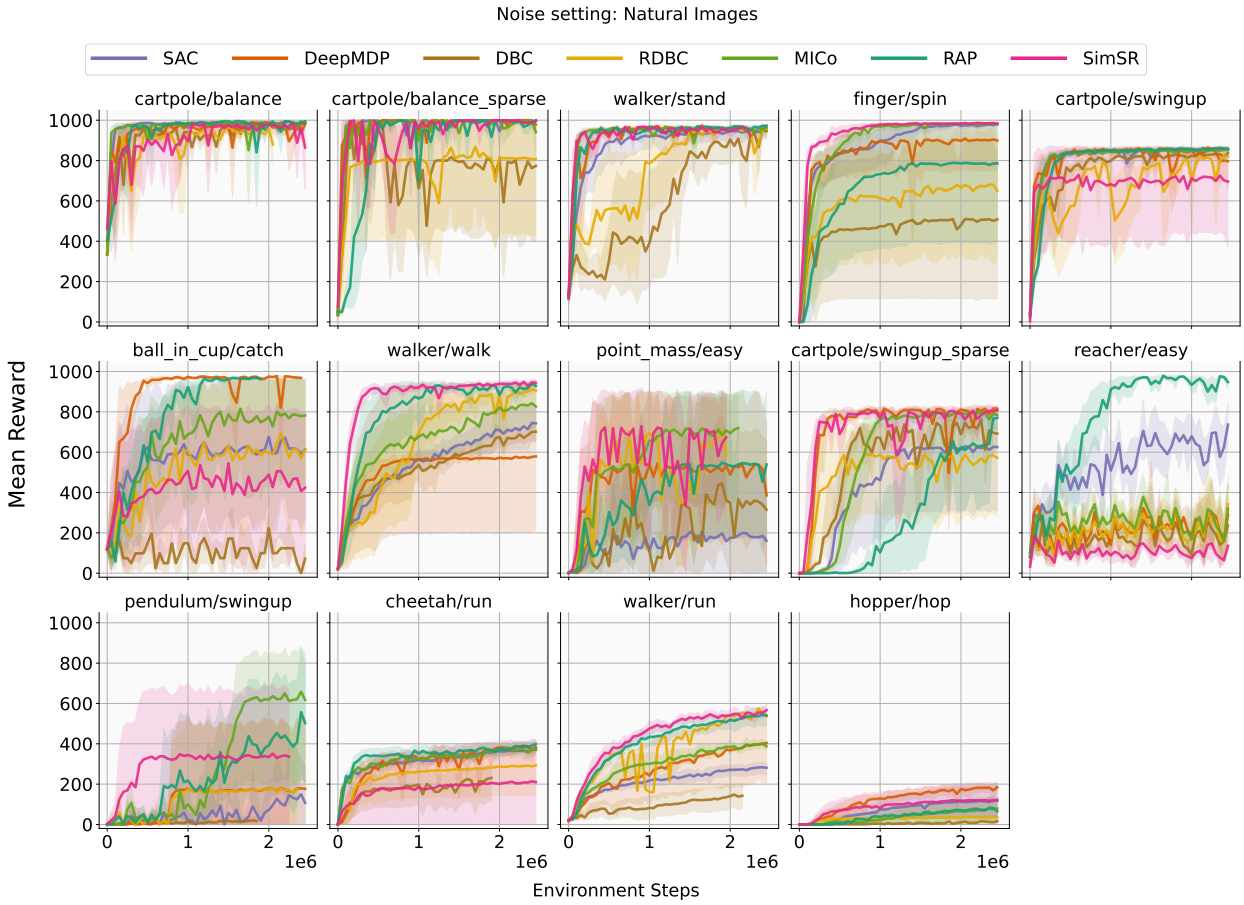

Figure 21: Individual pixel-based task performance.

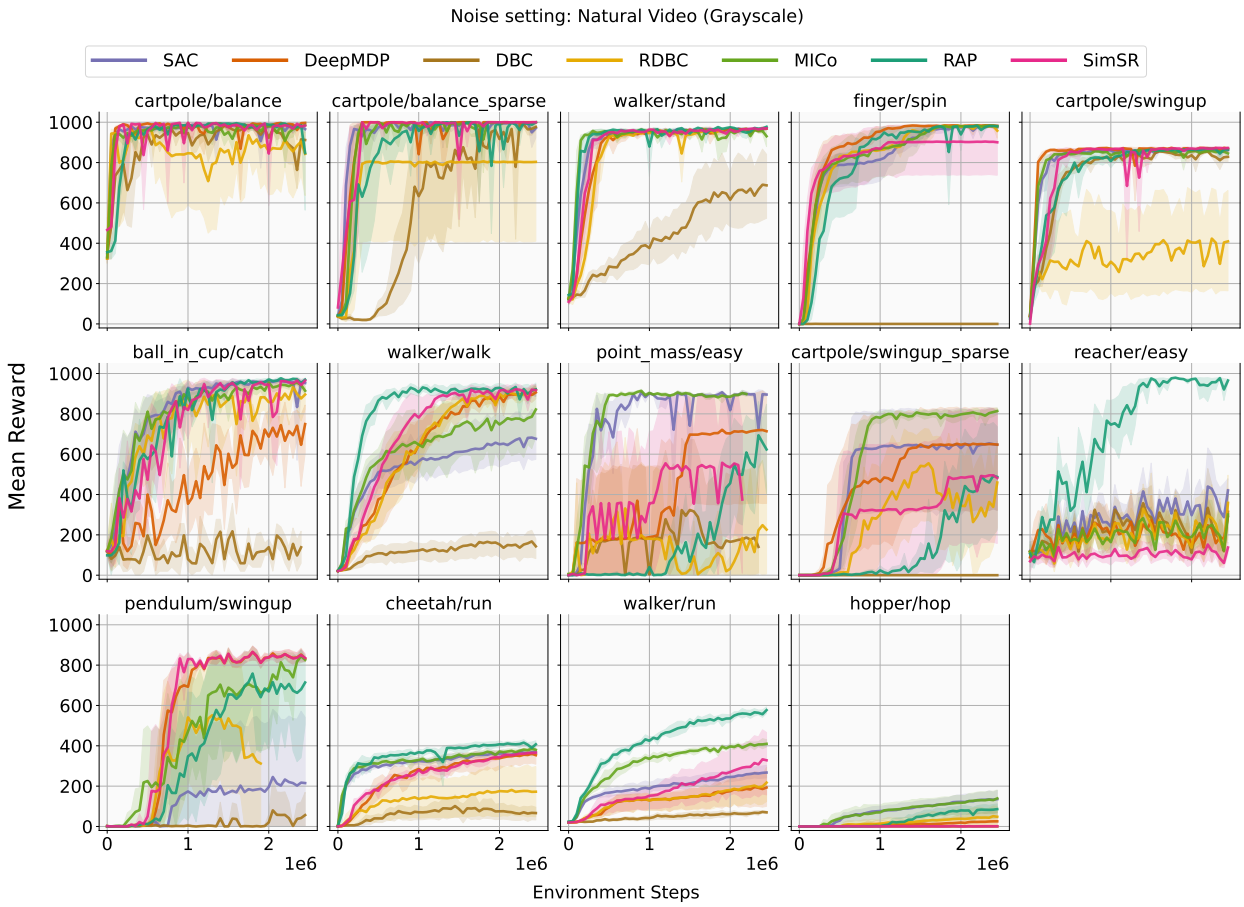

Figure 22: Individual pixel-based task performance.

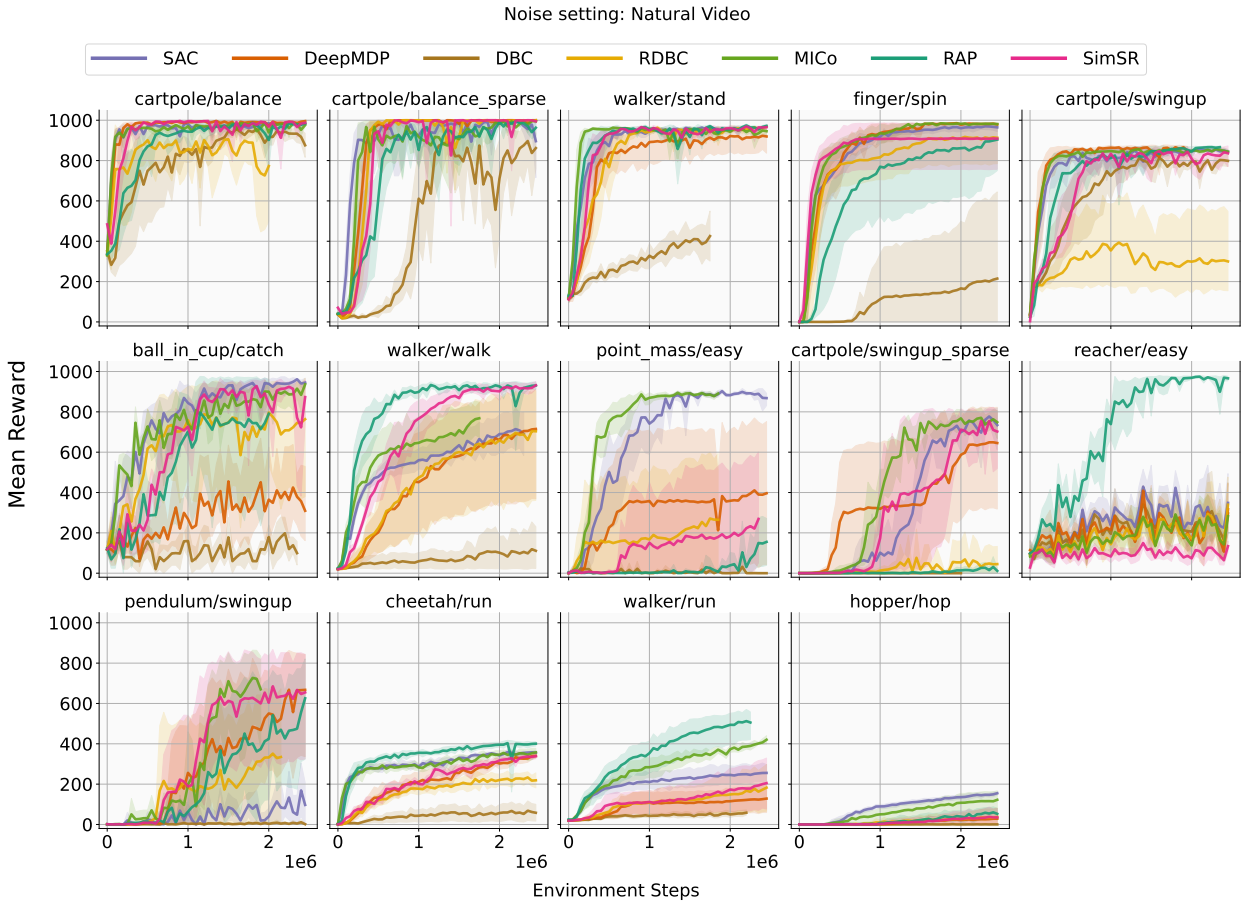

Figure 23: Individual pixel-based task performance.

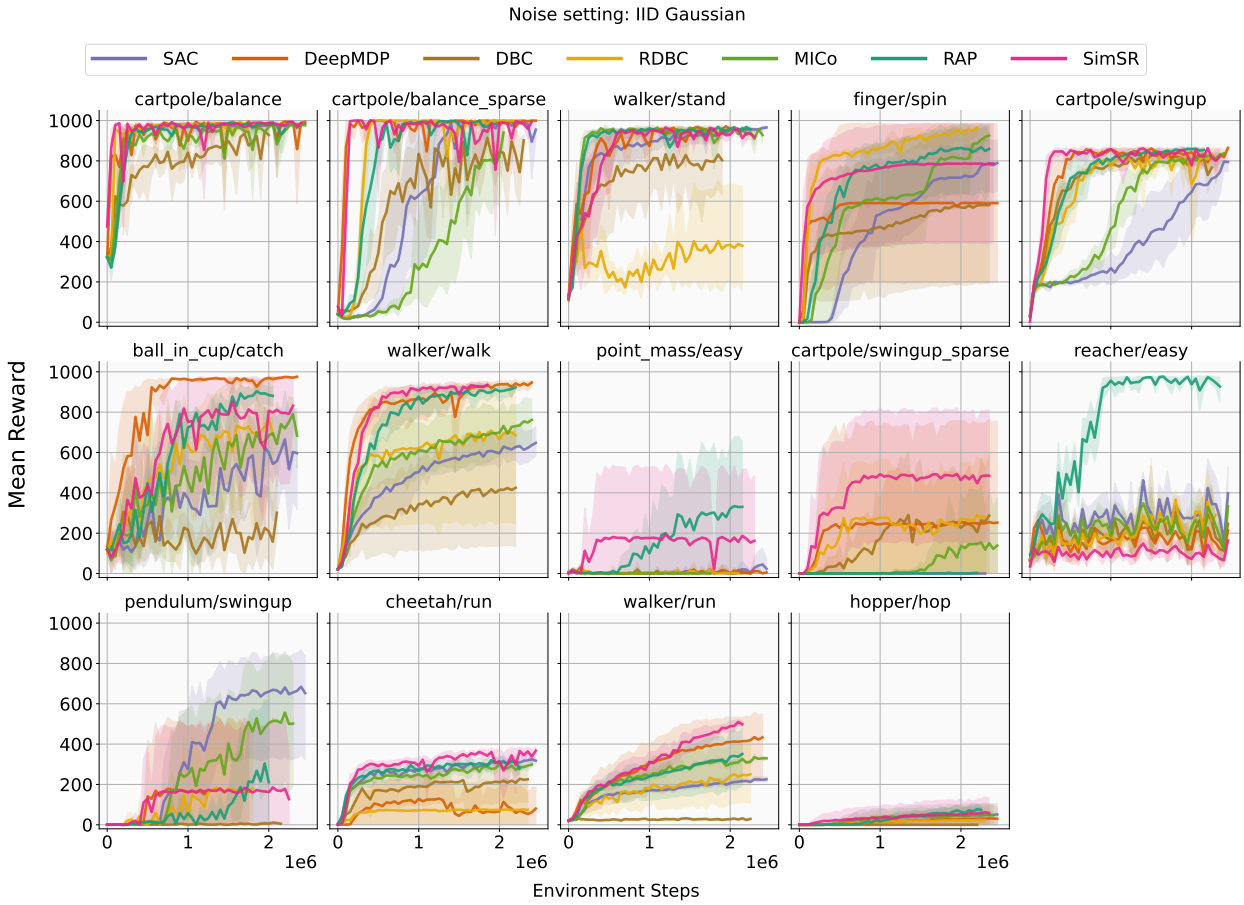

Figure 24: Individual pixel-based task performance.

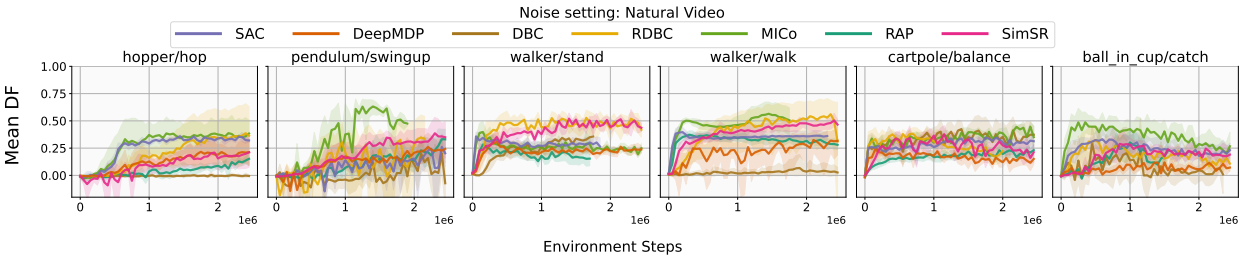

Figure 25: Denoising factor (ODD evaluation setting) on agent encoder ϕ.

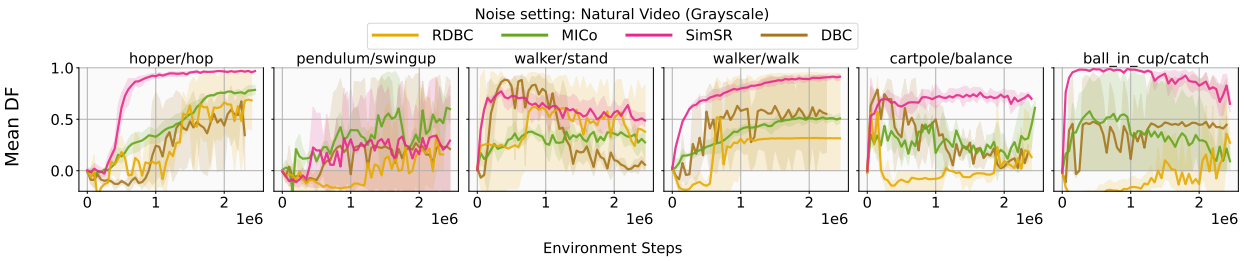

Figure 26: Denoising factor (ID evaluation setting) on isolated metric encoder $\tilde{\phi}$.

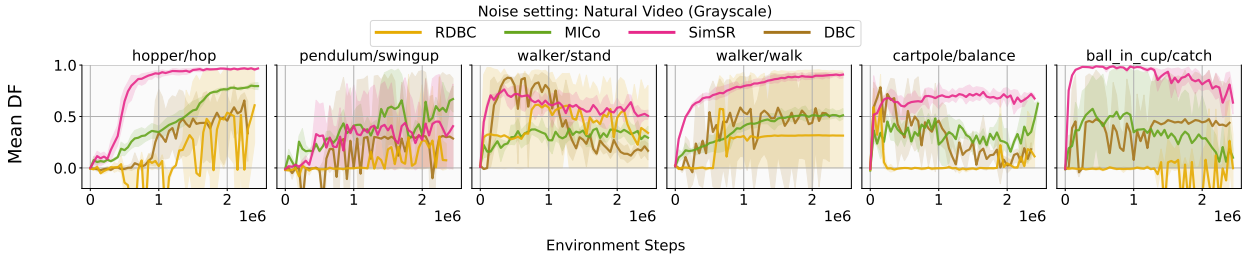

Figure 27: Denoising factor (OOD evaluation setting) on isolated metric encoder $\tilde{\phi}$.

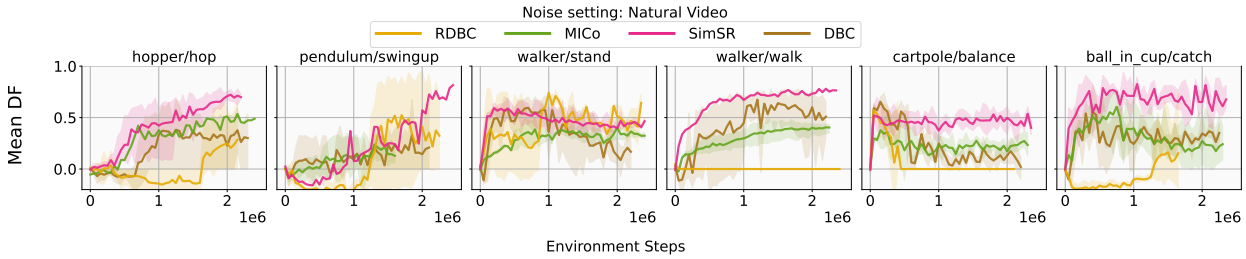

Figure 28: Denoising factor (ID evaluation setting) on isolated metric encoder $\tilde{\phi}$.

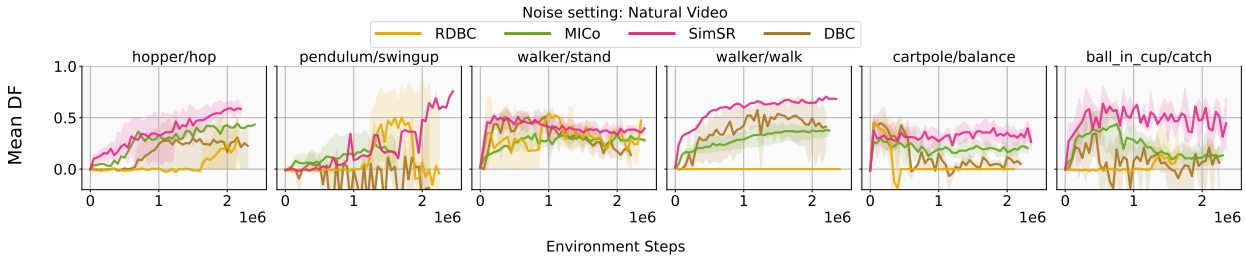

Figure 29: Denoising factor (OOD evaluation setting) on isolated metric encoder $\tilde{\phi}$.