Understanding Behavioral Metric Learning:

A Large-Scale Study on Distracting Reinforcement Learning Environments

Ziyan "Ray" Luo, Tianwei Ni, Pierre-Luc Bacon, Doina Precup, Xujie Si

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 embedding these learned distances in the representation space. While promising for robustness to task-irrelevant noise, as 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 reinforcement learning (RL), we evaluate five recent approaches, unified conceptually as isometric embeddings with varying design choices. We benchmark them with baselines across 20 state-based and 14 pixel-based tasks, spanning *370 task configurations* with diverse noise settings. Beyond final returns, we introduce the evaluation of a *denoising factor* to quantify the encoder's ability to filter distractions. To further isolate the effect of metric learning, we propose and evaluate an *isolated metric estimation* setting, in which the encoder is influenced solely by the metric loss. 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 recent 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. We adopt most hyperparameters provided in their codebases, as we believe they are well-tuned for their benchmarks that share the same denoised states as ours.
   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 Behavioral Metric Learning:

A Large-Scale Study on Distracting Reinforcement Learning Environments

Ziyan "Ray" Luo[1,2], **Tianwei Ni**[1,3],
Pierre-Luc Bacon[1,3,5], **Doina Precup**[1,2,5], **Xujie Si**[1,4,5]
ziyan.luo@mail.mcgill.ca, twni2016@gmail.com

[1]**Mila - Quebec Artificial Intelligence Institute** [2]**McGill University** [3]**Université de Montréal**
[4]**University of Toronto & Vector Institute** [5]**Canada CIFAR AI Chair**

Abstract

A key approach to state abstraction is approximating *behavioral metrics* (notably, bisimulation metrics) in the observation space and embedding these learned distances in the representation space. While promising for robustness to task-irrelevant noise, as 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 reinforcement learning (RL), we evaluate five recent approaches, unified conceptually as isometric embeddings with varying design choices. We benchmark them with baselines across 20 state-based and 14 pixel-based tasks, spanning *370 task configurations*[1] with diverse noise settings. Beyond final returns, we introduce the evaluation of a *denoising factor* to quantify the encoder's ability to filter distractions. To further isolate the effect of metric learning, we propose and evaluate an *isolated metric estimation* setting, in which the encoder is influenced solely by the metric loss. Finally, we release an open-source, modular codebase to improve reproducibility and support future research on metric learning in deep RL.[2]

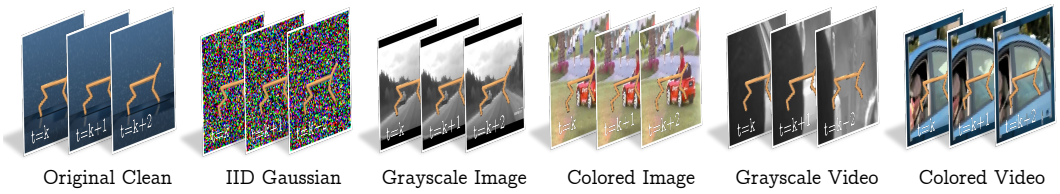

Figure 1: **Examples of background noise settings in pixel-based domains.** In image settings, the background is fixed; in video settings, it varies slightly; IID Gaussian noise is independently sampled each timestep.

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 (e.g., Fig. 1). 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

[1]200 state-based IID Gaussian (20 tasks × 10 noises), 84 pixel-based ID generalization (14 tasks × 6 noises), 30 state-based IID Gaussian with random projection (6 tasks × 5 noises), and 56 pixel-based OOD generalization (14 tasks × 4 noises).

[2]The artifact is available at https://github.com/Rayluo-mila/understanding-metric-learning.

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 into the broader class of **behavioral metrics** (Castro et al., 2023), which quantify state 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 a lack of systematic evaluation. First, metric learning's effectiveness relies on accurate estimation in theory, but it is challenging in practice due to many design choices. Moreover, prior work primarily measures performance through *returns*, without directly assessing the quality of 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 metric learning's impact on 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.

Contributions. In this paper, we provide an understanding of **how metric learning works in deep RL** through a systematic large-scale study. Our main contributions are as follows:

1. **Conceptual insights** (Sec. 3): We unify five recent metric learning approaches under an isometric embedding framework to identify key design choices. We analyze why some exact behavioral metrics provide a theoretical denoising guarantee, whereas others may not.
2. **Evaluation designs on denoising** (Sec. 4): To ensure a rigorous and comprehensive evaluation, we introduce diverse distraction benchmarks with varying difficulty levels, 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 by introducing the *denoising factor* (DF). Finally, we propose an *isolated metric estimation* setting (referred to as the isolated setting) to assess metric learning's contribution to denoising, independent of other losses.
3. **Comprehensive evaluation** (Sec. 5): We conduct a comprehensive benchmark of five metric learning methods and baselines across 20 state-based tasks with 10 IID Gaussian noise levels, and 14 pixel-based tasks with 6 distraction types, in DeepMind Control suite (Tassa et al., 2018), evaluating both return and DF. Beyond performance comparison, we assess the difficulty of our distracting benchmarks, and conduct targeted case studies to identify key design choices and examine the connection between metric learning and denoising throughout ablation study and isolated metric estimation.
4. **Open-source codebase** (footnote 2): We open-source a modular and efficient codebase to enhance the reproducibility and extensibility of metric-based methods in the RL community.

Main Findings. Based on the evaluation, we highlight the main findings as follows:

1. **Benchmarking results** (Sec. 5.1): SimSR, despite being designed for pixel tasks, outperforms all methods in return and denoising on state-based domains with IID Gaussian noise. RAP performs best in pixel-based tasks. Surprisingly, SAC and DeepMDP are strong baselines. Interestingly, common distractions like varying noise dimensions and grayscale videos add little difficulty, while random projection (state-based) and pixel Gaussian noise remain challenging.
2. **Case study insights** (Sec. 5.2): Further analysis reveals that SimSR's success on state-based tasks is largely driven by its use of self-prediction loss and feature normalization. Additionally, applying layer normalization tends to improve both return and denoising across all methods.
3. **Isolated setting results** (Sec. 5.3): When comparing DFs in the isolated setting, we find that the standalone benefit of learning metric by an explicit metric loss becomes marginal.

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). This formulation (i) encompasses many distracting environments in prior work, (ii) retains the generality of standard MDPs, (iii) exactly characterizes problems solvable by bisimulation metrics, and (iv) permits concise theoretical analysis and straightforward experiment design. 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*: $\forall\, z_1, z_2 \in \mathcal{Z},\ z_1 \neq z_2 \implies \operatorname{supp}(q(\cdot \mid z_1)) \cap \operatorname{supp}(q(\cdot \mid z_2)) = \emptyset$. 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: $\max_\pi \mathbb{E}_\pi \big[\sum_{t=0}^{\infty} \gamma^t R(q^{-1}(x_t), a_t)\big]$. 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)[3], 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 $R(s, a)$. EX-BMDPs guarantee the existence of a denoising map $D : \mathcal{Z} \to \mathcal{S}$ which 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)**[4], requires that the representation preserves reward information. The second, **self-prediction (ZP)**[5] (Ni et al., 2024), requires that the representation preserves latent dynamics information. Model-irrelevance abstraction thus defines compact yet informative encoders that retain essential information for optimal decision-

[3]From an MDP perspective, the grounded transition is $\mathcal{P}(x' \mid x, a) = \sum_{z' \in \mathcal{Z}} p(z' \mid q^{-1}(x), a) q(x' \mid z')$.

[4]Formally, in an EX-BMDP, RP condition is $\exists R_\kappa : \Psi \times \mathcal{A} \to \mathbb{R}$, s.t. $R(\phi^*(x), a) = R_\kappa(\phi(x), a), \forall x, a$.

[5]Formally, in an EX-BMDP, ZP condition is $\exists P_\nu : \Psi \times \mathcal{A} \to \Delta(\Psi)$, s.t. $\mathcal{P}(\psi' \mid x, a) = P_\nu(\psi' \mid \phi(x), a), \forall x, a, \psi'$, where $\mathcal{P}(\psi' \mid x, a) = \sum_{x' \in \mathcal{X}} \mathcal{P}(x' \mid x, a) \mathbf{1}(\phi(x') = \psi')$.

making (Subramanian et al., 2022). By definition, the RP and ZP conditions hold when $\phi = \phi^*$ and $\Psi = \mathcal{S}$.[6] This implies that the oracle encoder ϕ^* serves as a model-irrelevance abstraction.

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 backpropagation. The overall objective $J_{\text{DeepMDP}}(\phi)$ for the encoder in DeepMDP combines SAC loss with RP and ZP losses (Eq. 1, Fig. 9).

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 recent work instantiates it (Sec. 3.2). Please see Appendix Sec. C for background in metrics and metric learning and Sec. A for other related work.

3.1 Isometric Embedding: Between Behavioral Metrics and Representations

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)*[7]:

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**.*

The target metric captures differences in rewards and transition dynamics following a policy π. We omit the dependency on π for simplicity. The target metric is formulated as (Castro et al., 2023):

$$d_\mathcal{X}(x_1, x_2) := c_R d_R(x_1, x_2) + c_T d_T(d_\mathcal{X})(\mathcal{P}(x' \mid x_1), \mathcal{P}(x' \mid x_2)) \tag{3}$$

$$\approx c_R \hat{d}_R(r_1, r_2) + c_T \hat{d}_T(\hat{d}_\mathcal{X})(\hat{\mathcal{P}}(x' \mid x_1), \hat{\mathcal{P}}(x' \mid x_2)) = \hat{d}_\mathcal{X}(x_1, x_2), \tag{4}$$

where $r_1, r_2 \in \mathbb{R}$ are sampled immediate rewards based on x_1, x_2 following π and $\mathcal{P}(x' \mid x)$ is a next-state distribution following π. Here, d_R represents *immediate* state similarity by rewards and d_T is a probabilistic measure of *long-term* state similarity through transition distance, and $\hat{d}_R$ and $\hat{d}_T$ are approximants of d_R and d_T.

With isometric embedding assumption, we show the following lemma: for $x_1, x_2 \in \mathcal{X}$,

$$d_T(d_\mathcal{X})(\mathcal{P}(x' \mid x_1), \mathcal{P}(x' \mid x_2)) = d_T(d_\Psi)(\mathcal{P}_\phi(\psi' \mid x_1), \mathcal{P}_\phi(\psi' \mid x_2)), \tag{5}$$

where $\mathcal{P}_\phi(\psi' \mid x) = \sum_{x'} \mathcal{P}(x' \mid x) \mathbf{1}(\psi' = \phi(x'))$. The proof, provided in Appendix Sec. D.1, holds for all considered d_T. Intuitively, this result shows that isometry *preserves* transition distances in $\mathcal{X}$ when mapped to Ψ, a property implicitly assumed in prior work.

3.2 Design Choices in Behavioral Metric Learning

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

Method	$\hat{d}_R$	$\hat{d}_T$	d_Ψ	Metric Loss	Target Trick	Other Losses	Transition Model	Normali -zation
SAC (Haarnoja et al., 2018)	—	—	—	—	—	—	—	—
DeepMDP (Gelada et al., 2019)	—	—	—	—	—	RP + ZP	Probabilistic	—
DBC (Zhang et al., 2020)	Huber	W_2 closed-form	Huber	MSE	—	RP + ZP	Probabilistic	—
DBC-normed (Kemertas & Aumentado-Armstrong, 2021)	Huber	W_2 closed-form	Huber	MSE	—	RP + ZP	Deterministic	MaxNorm
MICo (Castro et al., 2021)	Abs.	Sample-based	Angular	Huber	✓	—	—	—
RAP (Chen & Pan, 2022)	RAP	W_2 closed-form	Angular	Huber	—	RP + ZP	Probabilistic	—
SimSR (Zang et al., 2022)	Abs.	Sample-based	Cosine	Huber	—	ZP	Prob. ensemble	L2Norm

[6] In this case, $R_\kappa(s, a) = R(s, a)$ and $P_\nu(s' \mid s, a) = p(s' \mid s, a)$.

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

Sec. 3.1 provides a general conceptual framework instantiated by several works through distinct design choices. Rather than detailing theoretical differences (Appendix Sec. C.2) and design choices defined in papers (Appendix Sec. C.3), we focus on *practical implementations* in their publicly available codebases, summarized in Table 1 and illustrated in Appendix Sec. F.2.

Choices of Target Metric $d_{\mathcal{X}}$. Methods vary on the choice of $d_{\mathcal{X}}$ (see Appendix Sec. C.2) and $\hat{d}_{\mathcal{X}}$ to approximate $d_{\mathcal{X}}$ – specifically, $\hat{d_R}$ and $\hat{d_T}$ that approximate d_R and d_T.

- $\hat{d_R}$: MICo and SimSR use absolute difference ("Abs." in Table 1, Eq. 20), DBC and DBC-normed use Huber distance ("Huber" in Table 1, Eq. 21), and RAP has a specific form (Appendix Sec. C.3).
- $\hat{d_T}$: To avoid expensive 1-Wasserstein computations in bisimulation metrics (Ferns et al., 2004), DBC, DBC-normed, and RAP approximate d_T using a Gaussian transition model with a 2-Wasserstein metric. In contrast, MICo and SimSR rely on sample-based distance approximations.

Choices of Representational Metric d_Ψ. To approximate d_Ψ, DBC and DBC-normed employ a Huber distance (a surrogate for L_2 distance); MICo, SimSR, and RAP use an angular distance.

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

$$J_M(\phi) = \ell\left(d_\Psi(\phi(x_1), \phi(x_2)) - \hat{d}_{\mathcal{X}}(x_1, x_2)\right), \tag{6}$$

where $\hat{d}_{\mathcal{X}}(x_1, x_2) = c_R \hat{d}_R(r_1, r_2) + c_T \hat{d}_T(d_\Psi)(\hat{\mathcal{P}}(\psi' \mid x_1), \hat{\mathcal{P}}(\psi' \mid x_2))$ derived by Eq. 3 and Eq. 5. Here, ℓ is Huber loss (Huber, 1992) in MICo, RAP and SimSR, or mean square error (MSE) in DBC and DBC-normed. 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.

Self-prediction (ZP) and Reward Prediction (RP) Loss. As discussed, approximating $d_{\mathcal{X}}$ often requires a transition model, and methods adopt distinct approaches: probabilistic models (DBC), ensembles of probabilistic models (SimSR), and deterministic models (DBC-normed). MICo, in contrast, employs a sample-based target metric that is free of ZP. Since all the listed metric learning methods use sampled immediate rewards to approximate d_R, an explicit reward model is not strictly necessary. However, following the convention of DeepMDP, in DBC, DBC-normed, and RAP, the RP loss is employed to further shape the representation.

Normalization in the Representation Space Ψ. DBC-normed employs max normalization to enforce boundedness, leveraging prior knowledge of value range constraints on target metrics. While SimSR requires L_2 normalization to enforce unit-length representations, all the other methods use LayerNorm (Ba et al., 2016) in pixel-based encoders. See Appendix Sec. C.4 for details.

3.3 Candidate Methods

We present the design choices of methods to be benchmarked in Table 1. In our experiment (Sec. 5), we follow DBC's implementation on DeepMDP which employs exponential moving average of ZP target (Ni et al., 2024) and excludes observation reconstruction loss. For DBC-normed, we exclude their additional components related to intrinsic rewards and inverse dynamics. For DBC-normed and SimSR, we replace their original transition models with a single probabilistic transition model. These modifications ensure that our study focuses on the effect of metric learning itself.

3.4 Why do Metrics (Not) Help with Denoising?

First, we define *denoising* as a form of generalization in which task-irrelevant noise is removed from observations, allowing a model to generalize across observations with unseen noise. Formally:

> An encoder ϕ is said to achieve **perfect denoising** in a EX-BMDP if, for any triplet $x, x_+, x_- \in \mathcal{X}$ such that $\phi^*(x) = \phi^*(x_+) \neq \phi^*(x_-)$, it holds that $\phi(x) = \phi(x_+) \neq \phi(x_-)$. That is, ϕ replicates the abstraction behavior of the oracle encoder ϕ^*.

We then discuss on the connection between denoising and target metric $d_{\mathcal{X}}$, which motivates further empirical investigation into whether metric learning facilitates denoising.

Metrics potentially help with denoising. Bisimulation metric (BSM, Def. 5) (Ferns et al., 2004; 2011) has perfect denoising in a EX-BMDP: for observations $x, x_+ \in \mathcal{X}$, $d_{\mathcal{X}}(x, x_+) = 0$ (see Appendix Sec. D.2 for a proof). Through Eq. 2, zero d_Ψ is ensured and the two observations are assigned to the same representation (Appendix Sec. C.1, Metric definition, (1)). PBSM (Def. 6) has a denoising property when the policy is *exo-free* (Islam et al., 2022) (see Appendix Sec. D.2 for a proof). Generally, the MICo distance (Def. 7) does not assign a zero distance to such x, x_+ (Castro et al., 2021) unless both the policy and transition function are deterministic, but empirical evidence indicates its potential to help with denoising (Chen & Pan, 2022; Zang et al., 2022).

Approximated metrics may not help with denoising. Although BSM has perfect denoising, it is inherently challenging to approximate (Castro, 2020). As a result, all of our candidate methods are based on PBSM (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021) or MICo (Castro et al., 2021; Zang et al., 2022; Chen & Pan, 2022). However, PBSM does not guarantee denoising observations under arbitrary policies, even when the policy is optimal (see Appendix Sec. D.2 for a detailed discussion). Furthermore, several gaps between theory and practice ($d_{\mathcal{X}}$ and $\hat{d}_{\mathcal{X}}$) exacerbate their denoising properties. Firstly, both PBSM and MICo are on-policy metrics, but the sampled rewards used in $\hat{d}_R$ (Sec. 3.2) are from a replay buffer (Eq. 15), which are off-policy. Secondly, the methods use approximated transition models, where an approximation error is introduced (Kemertas & Aumentado-Armstrong (2021), Appendix Sec. D). Thirdly, in the line of work that leverages behavioral metrics in deep RL, the metric loss is not the sole factor shaping the representation. The interplay among the metric loss, ZP loss, and critic loss can lead to undesirable outcomes.

4 Study Design on Metric Learning: Noise and 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 in limited settings* 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 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 introduce four noise settings under the EX-BMDP framework (Sec. 2.1), where observations follow $x \sim q(\cdot \mid z)$ with $z = (s, \xi)$, each designed to reflect distinct forms of environmental variation. IID Gaussian noise is applied to both state-based and pixel-based domains to simulate sensor-level randomness. In the state-based setting, we further consider a more challenging variant with random projection, inspired by information security (Dwork, 2006; Gentry, 2009), where data is privacy-preserving yet remains recoverable via decryption. Natural image and video settings apply only to pixel-based domains, simulating real-world background shifts in visual environments. The grayscale setting is widely adopted in metric learning (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021; Zang et al., 2022; Chen & Pan, 2022).[8] An illustration of the pixel-based noise settings is shown in Fig. 1.

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

[8] Some prior work present backgrounds in color but actually use grayscale in experiments, which may cause confusion.

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's foreground pixels, with q as a rendering function.

IID Gaussian Noise with Random Projection. It applies only to state-based domains where $s \in \mathbb{R}^n$. During initialization, a full-rank square matrix $\mathbf{A} \in \mathbb{R}^{(n+m)\times(n+m)}$ is constructed 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 $\xi_t \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2\mathbf{I})$ 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 of 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.

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 pixel-based noise setting replaces the clean background with *grayscale* or *colored* natural videos. 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.

4.2 Denoising Involves ID and OOD Generalization

The evaluation settings differ based on whether the *noise distribution* remains unchanged or shifts between training and testing. In **in-distribution (ID) generalization evaluation** setting, 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. For the **out-of-distribution (OOD) generalization evaluation setting**, the training and testing EX-BMDPs share the same task-relevant parts (i.e., $p(s' \mid s, a)$, $p(s_0)$, $\mathcal{R}(s, a)$) 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 OOD 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.[9] 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* under a policy π, then constructing a *positive example* x_+ that shares the same task-relevant state, i.e., $\phi^*(x) = \phi^*(x_+)$ (implying x and x_+ are bisimilar), and a *negative example* x_- for which this equality does not necessarily hold. We define DF under a specific policy π because an agent may lack access to anchors from other policies, and has no reason to denoise observations outside its training distribution.

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

$$\mathrm{Pos}^{\pi}_{d_\Psi}(\phi) := \mathbb{E}_{x\sim\rho_\pi(x),\xi_+\sim\rho(\xi_+),x_+\sim q(\cdot\mid\phi^*(x),\xi_+)}[d_\Psi(\phi(x), \phi(x_+))], \tag{7}$$

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 $\{0, 1, \ldots, N-1\}$.

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

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

$$\mathrm{Neg}^{\pi}_{d_\Psi}(\phi) := \mathbb{E}_{x,x_- \overset{\text{IID}}{\sim} \rho_\pi}[d_\Psi(\phi(x), \phi(x_-))]. \tag{8}$$

Definition 4 (Denoising factor (DF))**.** *The denoising factor of an encoder ϕ w.r.t. the metric d_Ψ is defined as the normalized difference between the negative and positive scores:*

$$\mathrm{DF}^{\pi}_{d_\Psi}(\phi) := \frac{\mathrm{Neg}^{\pi}_{d_\Psi}(\phi) - \mathrm{Pos}^{\pi}_{d_\Psi}(\phi)}{\mathrm{Neg}^{\pi}_{d_\Psi}(\phi) + \mathrm{Pos}^{\pi}_{d_\Psi}(\phi)} \in [-1, 1]. \tag{9}$$

DF measures denoising, with values above 0 indicating smaller distances for positive over negative pairs; higher values imply better denoising. The oracle encoder ϕ^* attains the maximum of 1.

4.4 Decoupling Metric Learning from RL for Denoising Evaluation

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 $J_{\mathrm{M}}(\phi)$ (Eq. 6). 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 an encoder and a policy. Although DFs under different policies may offer initial quantitative insights, such comparisons are not rigorous, as each reflects denoising ability on policy-specific data. Notably, policies that frequently revisit similar task-relevant states under varying noise can significantly inflate DF. Due to the above reasons, we propose to evaluate behavioral metric learning algorithms in an *isolated metric estimation* setting.

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 RL 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 collects data for learning the metrics (illustration see Fig. 11). This allows for a fair comparison of $\mathrm{DF}^{\pi}_{d_\Psi}(\tilde{\phi})$ 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. This setting can be naturally extended to cases where $\tilde{\phi}$ is optimized by a different combination of objectives than those used to optimize ϕ, for example, using J_{RP} and J_{ZP} to optimize $\tilde{\phi}$ while using J_{SAC} to optimize ϕ.

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 7) and **14 pixel-based** DMC tasks (listed in Table 8), under various noise settings with ID generalization evaluation. Our results (Sec. 5.1) offer a broad assessment of agent performance and task difficulty across a significantly larger set of tasks and noise settings than prior work. Based on these findings, we select a subset of representative tasks for case studies (Sec. 5.2) to identify key design choices (Sec. 3.2), and further investigate the isolated metric estimation setting (Sec. 4.4) in Sec. 5.3. OOD generalization following prior work is assessed in Sec. 5.4.

Evaluation Protocol. For aggregated scores, we report the *mean episodic reward* rather than the IQM (Agarwal et al., 2021b) to avoid ignoring tasks that are too easy or too challenging. Corresponding per-task results for all aggregated scores in the main text are provided in Appendix Sec. G. In our tables, each run's mean episodic reward, bounded within $[0, 1000]$, is computed as the average of 10 evaluation points collected between 1.95M and 2.05M steps, and then aggregated over seeds. All figures and tables display 95% confidence intervals (CIs) across tasks. We use 12 seeds for state-based and 5 seeds for pixel-based environments per task-noise combination.

Hyperparameters. We adopt the original hyperparameters from the referenced implementations (details and exceptions see Appendix Sec. F.1). The hyperparameters are widely used and considered well-tuned for pixel-based DMC with grayscale and clean backgrounds. Our settings retain the same

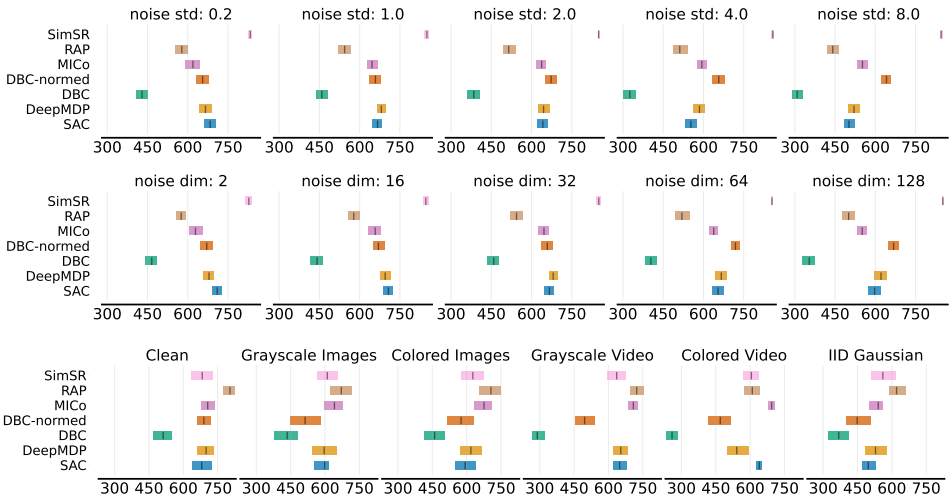

Figure 2: **Benchmarking results: performance of seven methods across diverse noise settings**, aggregating episodic rewards from **20 state-based** (first two rows) and **14 pixel-based** tasks (last row). "Noise std" denotes the IID Gaussian noise's standard deviation σ, while "noise dim" denotes its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

task-relevant state, varying only the noise type, which justifies this choice. Following DBC-normed, we use the same key hyperparameters for both pixel- and state-based tasks.

Approximation of DF (Eq. 9). Observations collected in the evaluation stage are considered *anchors*. We sample 16 positive and 16 negative examples for each anchor using the strategy in Sec. 4.3. We report $\mathrm{DF}^{\pi}_{||\cdot||_2}(\phi)$, where π denotes the agent's policy up to each evaluation point.

5.1 Benchmarking Methods on Various Noise Settings

Settings. For state-based DMC tasks, we apply IID Gaussian noise ($\boldsymbol{\mu} = \mathbf{0}$), varying either (a) standard deviations $\sigma \in \{0.2, 1.0, 2.0, 4.0, 8.0\}$ (with a fixed $m = 32$), or (b) noise dimensions $m \in \{2, 16, 32, 64, 128\}$ (with a fixed $\sigma = 1.0$). 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. 12, respectively. Per-task results are listed in Appendix Sec. G.

Implementation Details. For state-based tasks, the encoder is a three-layer MLP, as used by SAC and DBC-normed. 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 identical probabilistic latent transition models and reward models used in DBC and DBC-normed if applicable.

Benchmarking Findings. We summarize the key findings from Fig. 2 and Table 12.

- **SimSR** consistently achieves the highest performance in most state-based tasks, excelling in both return and DF. **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**, although not metric learning methods, deliver decent performance on both pixel-based and state-based tasks, but are often overlooked in prior work. Conversely, **DBC**, a commonly used metric learning baseline, consistently performs the worst among all methods.
- Within the ranges we tested in state-based tasks, both increasing the number of noise dimensions (at fixed $\sigma = 1.0$) and the noise standard deviation (at fixed $m = 32$) causes moderate reward drops. Well-performing methods remain robust to noise variation in both reward and DF (Fig. 12).
- In pixel-based domains, *grayscale natural video*, widely used in prior work, is not significantly harder than the clean background setting (e.g., for SAC and DeepMDP). Surprisingly, the *IID Gaussian noise* setting is the most challenging, warranting further study.

Table 2: Performance comparison without (R) and with LayerNorm (R'). The cell backgrounds in R' rows reflect $R' - R$: **red** if $R' - R > 0$, **blue** if $R' - R < 0$, with darker shades for larger magnitude.

Task		Methods						
		SAC	DeepMDP	DBC	DBC-normed	MICo	RAP	SimSR
cartpole/balance	R	967.5±12.3	928.7±32.3	814.1±86.6	973.7±12.4	966.6±9.2	950.3±71.2	999.5±0.5
	R'	979.5±20.1	994.6±3.6	943.6±24.1	975.5±19.9	936.1±29.8	981.7±19.1	980.2±19.3
finger/turn_easy	R	592.9±176.6	327.3±88.5	201.9±38.5	619.0±35.1	419.0±75.9	240.6±36.4	926.8±10.9
	R'	770.6±65.8	955.0±7.1	193.7±22.2	577.5±33.7	745.3±47.6	412.8±39.3	934.6±16.0
walker/run	R	635.3±19.8	347.8±84.0	23.9±2.6	628.9±25.7	455.9±41.3	649.4±11.1	760.6±19.4
	R'	534.5±53.6	776.0±5.9	342.9±54.5	759.8±19.4	611.0±22.5	661.6±88.4	761.6±20.0
quadruped/run	R	233.8±59.0	381.1±64.9	219.5±63.5	433.3±47.3	417.9±44.2	441.1±93.7	847.4±21.7
	R'	483.8±6.0	891.1±17.8	291.3±55.0	509.5±35.4	467.4±21.8	687.3±59.8	832.9±63.4
finger/turn_hard	R	177.6±66.1	168.3±50.4	97.9±11.8	414.7±49.5	207.2±53.8	110.8±17.0	885.4±24.5
	R'	495.7±53.1	925.8±14.7	95.9±12.4	473.4±39.9	335.1±42.6	201.1±26.3	917.1±13.9
hopper/hop	R	0.1±0.0	31.3±16.7	0.3±0.3	51.1±13.4	0.4±0.3	0.8±0.5	233.9±22.6
	R'	12.4±4.9	195.4±19.9	6.2±4.8	125.8±22.3	1.8±2.0	1.0±0.3	207.4±36.4

- Different algorithms excel in different tasks (Table 7, Table 8), e.g., RAP in reacher/easy, MICo in point_mass/easy (Fig. 26). Broad task coverage is essential to ensure generalizable insights.
- Adding objectives trades off computation efficiency. As shown in Table 12, the time cost of optimizing 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, we conduct case studies on the design choices outlined in Sec. 3.2. Six *easy-to-hard* state-based DMC tasks (see Table 2) are selected for detailed analysis.

Case studies design. 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 effectiveness (Zang et al., 2022). This inspires us to examine whether normalization benefits *other* metric learning methods. Considering the target metric *misspecification* issue (see Appendix Sec. C.4), we examine the effect of incorporating LayerNorm on the representation space rather than using L_2 normalization in methods that do not inherently require it. Second, several techniques used by the best-performing methods merit further analysis. Specifically, 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). To evaluate the effectiveness of these techniques, we apply them to DBC-normed, which originally does not include any of these modifications. Third, we investigate the performance of methods *with LayerNorm* in a challenging setting: *IID Gaussian noise with random projection* (Sec. 4.1) with $\sigma \in \{0.2, 1.0, 2.0, 4.0, 8.0\}$ (with a fixed noise dimension $m = 32$), shown in Fig. 4. Important findings in Table 2, Fig. 28, Fig. 3, and Fig. 4 are as follows:

- **Most methods benefit from LayerNorm in the representation space**, improving both reward (Table 2) and DF (Fig. 29). Notably, **DeepMDP with LayerNorm performs comparably to SimSR**.[10] For DBC-normed, employing Huber loss for the metric and incorporating the target trick yield a modest performance improvement (Fig. 28).
- **ZP loss is crucial for SimSR's success** in noisy state-based tasks (Fig. 3).
- A significant performance drop occurs for all agents when increasing the noise standard deviation in **IID Gaussian with random projection** setting (Fig. 4, Fig. 30), even with LayerNorm applied. Nevertheless, DeepMDP and SimSR remain relatively robust to the noise.

5.3 Isolated Metric Estimation Setting: Does Metric Learning Help with Denoising?

We further analyze the proposed isolated setting (Sec. 4.4) in the 6 selected tasks (Fig. 5). The experiments include: (i) isolated metric estimation for metric learning methods, SAC, and DeepMDP where $\tilde{\phi}$ is optimized using either the Q loss, ZP loss alone (i.e., the minimalist algorithm (Ni et al., 2024)), or both RP and ZP losses. (ii) The same as (i) but with LayerNorm applied to $\tilde{\phi}$. (iii) Build-

[10] 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.

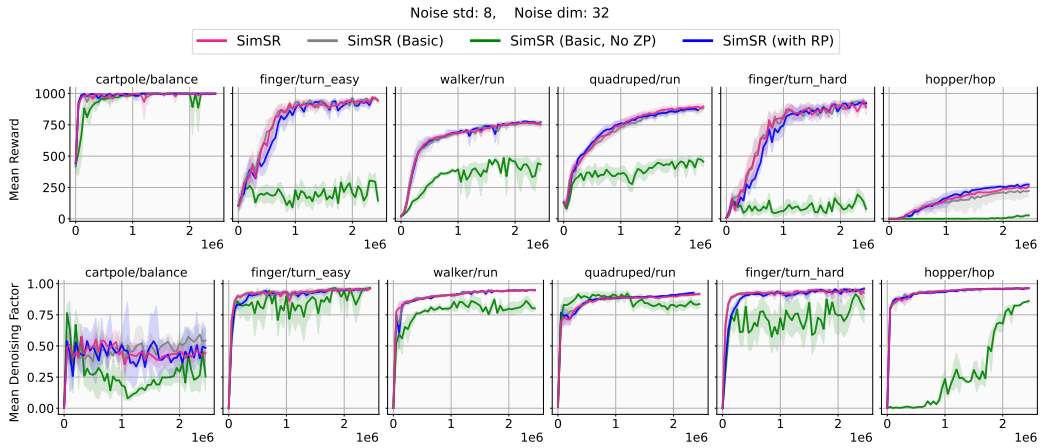

Figure 3: **Ablation study on ZP loss on SimSR.** "SimSR" is the agent benchmarked in Sec. 5.1, where ZP is integral to the metric estimation. Therefore, we resort to "SimSR (Basic)" setting (Theorem 2, Zang et al. (2022)), where ZP is independent of the metric estimation, and "SimSR (Basic, No ZP)" is the setting that ZP is detached from SimSR (Basic). "SimSR (with RP)" adds RP loss to original SimSR. This ablation highlights the impact of detaching ZP on the overall performance. X-axis stands for the environmental step.

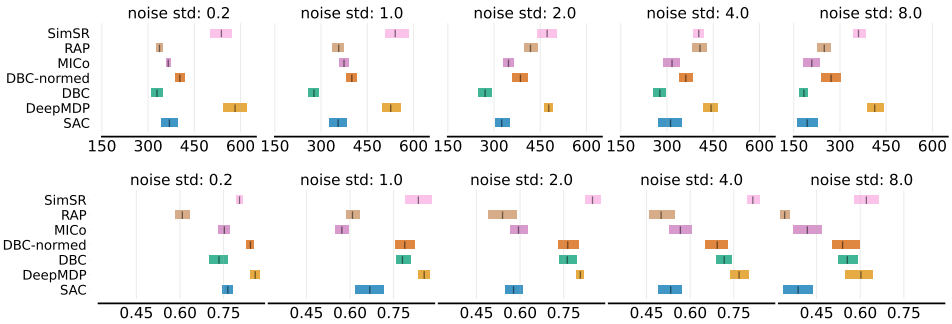

Figure 4: **Aggregated reward (top row) and DF (bottom row)** of seven agents on **IID Gaussian with random projection** settings, varying noise standard deviation, in the 6 selected state-based tasks.

ing on (ii), additionally applying ZP loss to $\tilde{\phi}$ for all metric learning methods. All evaluations are conducted on ID generalization using SAC with LayerNorm as the base agent. We observe that:

- Generally, metrics learned in isolation provide some denoising but underperform relative to optimizing the ZP loss on $\tilde{\phi}$ (i.e., DeepMDP (w/o RP) in Fig. 5, middle row). Including an additional RP loss to shape $\tilde{\phi}$ (as in DeepMDP) yields limited DF improvement to the ZP-only setting.
- Applying LayerNorm to the isolated encoder $\tilde{\phi}$ substantially improves DF for DeepMDP (with or w/o RP), but offers only modest gains for metric learning methods (top and middle rows of Fig. 5).
- Adding metric losses to ZP loss on $\tilde{\phi}$ generally does not improve DF (Fig. 5, mid and bottom rows).
- MICo's DF remains relatively low, which aligns with its theoretical property that the metric for positive examples is non-zero (Def. 7), as MICo does not enforce zero self-distance.

5.4 OOD Generalization Evaluation on Pixel-based Tasks

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. 7 denotes using grayscale videos as distracting backgrounds for both training and evaluation, with distinct video samples in each phase. Takeaways in Fig. 6 and Fig. 7 are as follows:

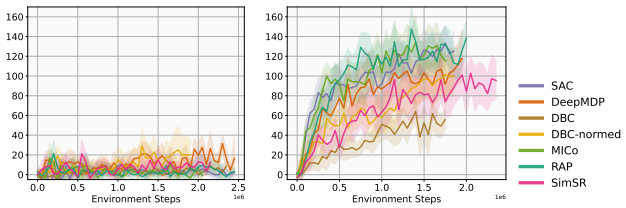

Figure 6: **Reward difference** between ID and OOD evaluation in the grayscale video setting (left) and colored video setting (right), aggregated over 14 pixel-based tasks in Table 8.

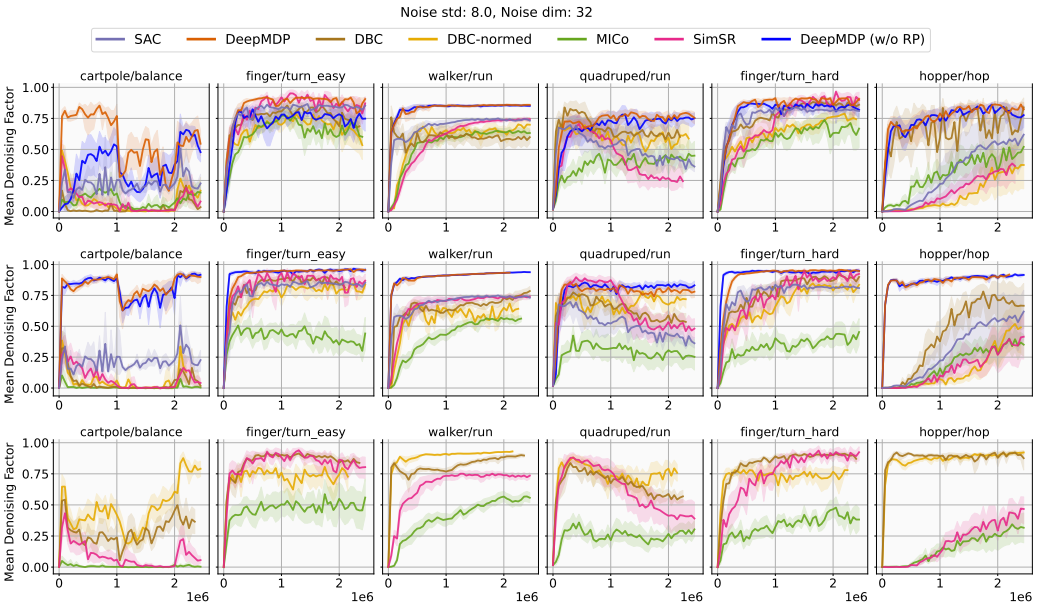

Figure 5: DF for isolated encoder $\tilde{\phi}$ – **top row**: without LayerNorm; **middle row**: with LayerNorm; **bottom row**: with LayerNorm, $\tilde{\phi}$ co-trained with metric and ZP losses. See Fig. 36 for reward curves and Fig. 37 for DF for the agent encoder ϕ co-trained with metric and RL losses.

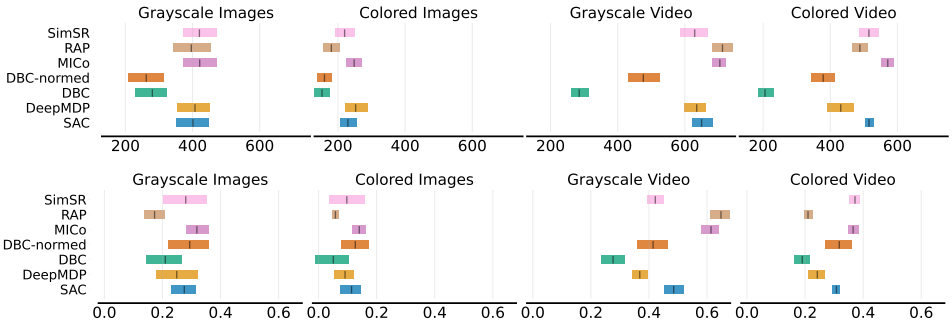

Figure 7: Aggregated reward (top row) and DF (bottom row) of 7 agents on various noise settings in 14 pixel-based tasks in Table 8 with **OOD generalization** evaluation.

- Comparing Fig. 7 (OOD) with Fig. 2 (ID), all methods struggle to generalize in both grayscale and colored image settings. Unlike video backgrounds, which provide changes over time that help distinguish relevant and irrelevant features, static images lack such variation, making adaptation to unseen backgrounds more difficult.
- Even with OOD generalization evaluation, SAC and DeepMDP remain competitive (Fig. 7).
- OOD generalization is significantly more challenging in the *colored* video setting than in the *grayscale* video setting (Fig. 6). Surprisingly, even baselines like SAC exhibit a low reward difference, questioning the necessity of incorporating a metric loss in the widely used grayscale setting.

6 Future Work

Our empirical study focuses on continuous control, particularly locomotion, with SAC as the base RL algorithm. As future work, we plan to extend this to discrete control domains and other embodiments with alternative base RL algorithms. To better disentangle algorithmic effects, future metric learning research may benefit from evaluation in *conceptually simple yet empirically challenging* environments with distracting noise, such as Gaussian noise with random projection in state-based domains (Sec. 4.1). In addition, evaluation could include the denoising factor (Sec. 4.3) and comparisons between in- and out-of-distribution generalization (Sec. 4.2). Building on our findings, future work could design new metric-based objectives that work well with self-prediction and normalization, providing additional benefits beyond what these components already offer.

Acknowledgments

Ziyan Luo is funded by the Canada CIFAR AI Chair Program. Tianwei Ni is funded by RBC Borealis Graduate Fellowship and Google DeepMind. We thank Sahand Rezaei-Shoshtari, Jianda Chen, Weijian Liao, Pablo Samuel Castro, Ayoub Echchahed, Lu Li, Xutong Zhao, Wesley Chung, Gandharv Patil, Aditya Mahajan, and anonymous reviewers for valuable discussions that helped shape our paper. This research was enabled by compute resources provided by Mila (https://mila.quebec) and the Digital Research Alliance of Canada (https://alliancecan.ca/).

References

Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*, 2021a. 19, 20

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, 2021b. 8

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5, 9, 20, 26

Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. *arXiv preprint arXiv:1902.05605*, 2019. 20

Ondrej Biza and Robert Platt. Online abstraction with mdp homomorphisms for deep learning. *arXiv preprint arXiv:1811.12929*, 2018. 19

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. 2, 6, 19, 21, 22, 28

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. 4, 5, 6, 19, 21, 23, 24, 27, 29

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. 2, 4, 19

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. 2, 4, 6, 7, 19, 20, 21, 23, 25, 29, 30

Jianda Chen, Zichen Chen, Sinno Pan, Tianwei Zhang, et al. State chrono representation for enhancing generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 37:73309–73336, 2024. 19

Robert Dadashi, Shideh Rezaeifar, Nino Vieillard, Léonard Hussenot, Olivier Pietquin, and Matthieu Geist. Offline reinforcement learning with pseudometric learning. In *International Conference on Machine Learning*, pp. 2307–2318. PMLR, 2021. 19

Josée Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled markov processes. *Information and Computation*, 179(2):163–193, 2002. 18

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. 3, 21

Cynthia Dwork. Differential privacy. In *International colloquium on automata, languages, and programming*, pp. 1–12. Springer, 2006. 6

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. 3, 21

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. 2, 5, 6, 18, 21, 22, 27

Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011. 2, 6, 18, 21, 22, 27

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. 20

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. 4, 19

Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 169–178, 2009. 6

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial intelligence*, 147(1-2):163–223, 2003. 1, 3, 18, 22

Jake Grigsby and Yanjun Qi. Measuring visual generalization in continuous control from pixels, 2020. 20

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33: 21271–21284, 2020. 19

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. 3, 4, 21

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *International Conference on Robotics and Automation*, 2021. 20

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. *arXiv preprint arXiv:2310.16828*, 2023. 20

Philippe Hansen-Estruch, Amy Zhang, Ashvin Nair, Patrick Yin, and Sergey Levine. Bisimulation makes analogies in goal-conditioned reinforcement learning. In *International Conference on Machine Learning*, pp. 8407–8426. PMLR, 2022. 19

Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. *arXiv preprint arXiv:2110.02034*, 2021. 20

Joey Hong, Anca Dragan, and Sergey Levine. Offline rl with observation histories: Analyzing and improving sample complexity. *arXiv preprint arXiv:2310.20663*, 2023. 19

Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics: Methodology and distribution*, pp. 492–518. Springer, 1992. 5, 24

Marcel Hussing, Claas Voelcker, Igor Gilitschenski, Amir-massoud Farahmand, and Eric Eaton. Dissecting deep rl with high update ratios: Combatting value divergence. *arXiv preprint arXiv:2403.05996*, 2024. 20

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. 3, 6, 28

Nan Jiang. Notes on state abstractions, 2018. 18

Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 31

Mete Kemertas and Tristan Aumentado-Armstrong. Towards robust bisimulation metric learning. *Advances in Neural Information Processing Systems*, 34:4764–4777, 2021. 2, 4, 6, 7, 8, 20, 21, 23, 25

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999. 3

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

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *AI&M*, 1(2):3, 2006. 1, 3, 18, 28, 29

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. 20

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. 2, 19

Weijian Liao, Zongzhang Zhang, and Yang Yu. Policy-independent behavioral metric-based representation for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 8746–8754, 2023. 19

Qiyuan Liu, Qi Zhou, Rui Yang, and Jie Wang. Robust representation learning by clustering with bisimulation metrics for visual reinforcement learning with distractions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 8843–8851, 2023. 19

Robin Milner. *A calculus of communicating systems*. Springer, 1980. 18

Vivek Myers, Chongyi Zheng, Anca Dragan, Sergey Levine, and Benjamin Eysenbach. Learning temporal distances: Contrastive successor features can provide a metric structure for decision-making. *arXiv preprint arXiv:2406.17098*, 2024. 21

Michal Nauman, Michał Bortkiewicz, Piotr Miłoś, Tomasz Trzciński, Mateusz Ostaszewski, and Marek Cygan. Overestimation, overfitting, and plasticity in actor-critic: the bitter lesson of reinforcement learning. *arXiv preprint arXiv:2403.00514*, 2024. 20

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. 1, 3, 5, 7, 10, 19, 20

Evgenii Nikishin, Romina Abachi, Rishabh Agarwal, and Pierre-Luc Bacon. Control-oriented model-based reinforcement learning with implicit differentiation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 7886–7894, 2022. 20

Prakash Panangaden. *Labelled Markov Processes*. Imperial College Press, 2009. 18

Prakash Panangaden, Sahand Rezaei-Shoshtari, Rosie Zhao, David Meger, and Doina Precup. Policy gradient methods in the presence of symmetries and state abstractions. *Journal of Machine Learning Research*, 25 (71):1–57, 2024. 19

David Park. Concurrency and automata on infinite sequences. In *Theoretical Computer Science: 5th GI-Conference Karlsruhe, March 23-25, 1981*, volume 104, pp. 167. Springer, 1981. 18

Brahma Pavse and Josiah Hanna. State-action similarity-based representations for off-policy evaluation. *Advances in Neural Information Processing Systems*, 36:42298–42329, 2023. 19

Balaraman Ravindran. *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst, 2004. 19

Balaraman Ravindran and Andrew G Barto. Model minimization in hierarchical reinforcement learning. In *Abstraction, Reformulation, and Approximation: 5th International Symposium, SARA 2002 Kananaskis, Alberta, Canada August 2–4, 2002 Proceedings 5*, pp. 196–211. Springer, 2002. 19

Sahand Rezaei-Shoshtari, Rosie Zhao, Prakash Panangaden, David Meger, and Doina Precup. Continuous mdp homomorphisms and homomorphic policy gradient. *Advances in Neural Information Processing Systems*, 35:20189–20204, 2022. 19

Max Rudolph, Caleb Chuck, Kevin Black, Misha Lvovsky, Scott Niekum, and Amy Zhang. Learning action-based representations using invariance. *arXiv preprint arXiv:2403.16369*, 2024. 21

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. 7

Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020. 19, 20

Yutaka Shimizu and Masayoshi Tomizuka. Bisimulation metric for model predictive control. *arXiv preprint arXiv:2410.04553*, 2024. 19

Satinder Singh, Tommi Jaakkola, and Michael Jordan. Reinforcement learning with soft state aggregation. *Advances in neural information processing systems*, 7, 1994. 1

Laura Smith, Ilya Kostrikov, and Sergey Levine. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022. 20

Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite–a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021. 20

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. 4

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. 2, 8, 21

Jonathan Taylor. Lax probabilistic bisimulation. 2008. 19

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. 2, 19

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. 8, 21

Elise Van der Pol, Thomas Kipf, Frans A Oliehoek, and Max Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. *arXiv preprint arXiv:2002.11963*, 2020a. 19

Elise Van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020b. 19

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. 20

Yiming Wang, Ming Yang, Renzhi Dong, Binbin Sun, Furui Liu, et al. Efficient potential-based exploration in reinforcement learning using inverse dynamic bisimulation metric. *Advances in Neural Information Processing Systems*, 36:38786–38797, 2023. 19, 20

Yiming Wang, Kaiyan Zhao, Furui Liu, et al. Rethinking exploration in reinforcement learning with effective metric-based exploration bonus. *Advances in Neural Information Processing Systems*, 37:57765–57792, 2024. 19

Ward Whitt. Approximations of dynamic programs, i. *Mathematics of Operations Research*, 3(3):231–243, 1978. 19

Ward Whitt. Approximations of dynamic programs, ii. *Mathematics of Operations Research*, 4(2):179–185, 1979. 19

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021a. 29

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. 9, 20, 21

Zhecheng Yuan, Sizhe Yang, Pu Hua, Can Chang, Kaizhe Hu, and Huazhe Xu. Rl-vigen: A reinforcement learning benchmark for visual generalization. *Advances in Neural Information Processing Systems*, 36: 6720–6747, 2023. 20

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. 1, 2, 4, 6, 7, 8, 10, 11, 19, 20, 21, 23, 24, 25, 26, 27, 29

Hongyu Zang, Xin Li, Leiji Zhang, Yang Liu, Baigui Sun, Riashat Islam, Remi Tachet des Combes, and Romain Laroche. Understanding and addressing the pitfalls of bisimulation-based representations in offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:28311–28340, 2023. 19

Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *arXiv preprint arXiv:1811.06032*, 2018. 20

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. 1, 2, 4, 6, 7, 8, 19, 20, 21, 23, 27, 31

Supplementary Materials

The following content was not necessarily subject to peer review.

Appendix

Table of Contents

A Related Work **18**

B Notation **21**

C Background on Metrics and Metric Learning **21**

C.1 Metric, Pseudometric, and Diffuse Metric 21

C.2 Definitions of Various Behavioral Metrics 22

C.3 Approximating the Behavioral Metrics 23

C.4 Normalization Techniques 25

D Proof **27**

D.1 Proof of Transition Distance Preservation under Isometric Embedding 27

D.2 Proof of Denoising Property of Bisimulation Metrics 27

E Difficulty Levels of the Tasks **29**

F Implementation Details **29**

F.1 Hyperparameters 29

F.2 Model Architecture 31

F.3 Pixel-based Environmental Setup 31

G Additional Experiment Results **33**

A Related Work

State Abstraction and Behavioral Metrics. State abstraction in MDPs is traditionally achieved by grouping equivalent states into a single abstract state. Various types of abstractions are proposed using the criteria of different equivalence relations on aggregating the states, e.g., bisimulation (model-irrelevance) abstraction and Q^*-irrelevance abstraction (Li et al., 2006; Jiang, 2018). Bisimulation, as a canonical equivalence relation, originates from concurrency theory in the context of labeled transition systems (Milner, 1980; Park, 1981) and labeled Markov process (Desharnais et al., 2002; Panangaden, 2009). In the context of MDPs, it can be regarded as a refinement of both action-sequence equivalence (*identical future reward sequences* given the same action sequence as input) and optimal value equivalence (Givan et al., 2003). To relax the strict dichotomy inherent in bisimulation relations, Ferns et al. (2004) propose using pseudometrics to measure the degree of bisimilarity between two states in a finite MDP and Ferns et al. (2011) extend it further to

continuous MDPs. Policy-dependent bisimulation metric (PBSM) (Castro, 2020) is a scalable variant of the bisimulation metric (BSM) that restricts behavioral similarity to the current policy, eliminating the need to evaluate all actions. GCB (Hansen-Estruch et al., 2022) extends PBSM to goal-conditioned RL, proposing a similar metric that measures the distance between two state-goal pairs. Castro et al. (2021) propose the MICo distance, which enables sample-based computation of transition distance while also offering theoretical benefits (Castro et al., 2023). Building on MICo, improvements have been introduced by SimSR (Zang et al., 2022) and RAP (Chen & Pan, 2022) (see Sec. C.2). Chen et al. (2024) further propose a multi-step behavioral metric that measures the distance between two pairs of states.

Beyond state abstraction, state-action abstraction has been extensively studied as another paradigm of model minimization. As early as 1978, Whitt (1978; 1979) proposes constructing approximate dynamic-programming models by restricting to subsets of states and actions, establishing bounds on how the approximate return function diverges from the true optimum. In modern RL, this abstraction paradigm manifests in the form of *MDP homomorphism* (Ravindran & Barto, 2002; Ravindran, 2004), which compresses state-action pairs under equivalence relations that preserve reward and transition dynamics, as well as in their relaxation via the *lax bisimulation metric* (Taylor, 2008). A promising application is the exploitation of environmental symmetries, and recent work has focused on scalable methods for discovering and approximating such homomorphisms (Biza & Platt, 2018; Van der Pol et al., 2020b;a; Rezaei-Shoshtari et al., 2022; Liao et al., 2023; Panangaden et al., 2024).

Representation and Metric Learning in RL. As shown in Table 1, many compared methods adopt self-prediction loss in addition to metric loss to shape the representation; see Ni et al. (2024) for a comprehensive review. Notably, DeepMDP (Gelada et al., 2019) provides a theoretically grounded and empirically validated framework for self-predictive representation learning in RL, incorporating ZP and RP auxiliary losses alongside standard RL objectives to construct a latent MDP. SPR (Schwarzer et al., 2020) adopts a similar self-supervised approach by learning a transition model and shaping the representation using signals from the k-th future observation rather than the next observation in DeepMDP.

There are multiple ways to connect representation and metric learning in RL. For example, in Sec. 3, we introduce isometric embedding as a unifying approach adopted by all our candidate methods. Beyond this, Agarwal et al. (2021a) relax the binary indicator in the contrastive loss by substituting it with a metric-induced continuous similarity measure. Liu et al. (2023) use metrics for prototype representation clustering, and Wang et al. (2023; 2024) utilize the learned metrics to shape the reward.

Beyond their use in online, model-free RL, behavioral metrics have also been applied to representation learning in offline RL (Dadashi et al., 2021; Hong et al., 2023; Pavse & Hanna, 2023; Zang et al., 2023) and model-based RL (Shimizu & Tomizuka, 2024).

Evaluation on Representation Learning and Denoising. A closely related work, Tomar et al. (2021), evaluate representation learning methods on six *distracting pixel-based* DMC tasks and Atari games. They show that a simple baseline using reward prediction and self-prediction (akin to DeepMDP in our implementation) outperforms one metric learning method (i.e., DBC). Similarly, Li et al. (2022) study self-supervised losses like BYOL (Grill et al., 2020) on six *clean pixel-based* DMC tasks and Atari games. They find that BYOL, akin to self-prediction, benefits model-free agents but is inferior to data augmentation. For denoising evaluation, previous works (Zhang et al., 2020; Chen & Pan, 2022; Zang et al., 2022) *qualitatively* assess it by visualizing representations in a two-dimensional space using t-SNE plot, showing their connection to the approximated state-value function. The learned encoder is evaluated by determining whether anchor-positive pairs are closer than anchor-negative pairs in these plots, although these results may not be statistically significant.

Unlike prior empirical studies, our work specifically evaluates metric learning (five methods) across a broader range of state-based and pixel-based DMC tasks, considering both denoising factors and returns. Beyond self-prediction, we identify other key design choices, such as layer normalization

and metric loss function, that impact performance. Denoising factors are computed from batches of random samples throughout evaluation.

Benchmarks on Denoising and Generalization. In state-based domains, IID Gaussian noises are commonly employed in prior work (Kemertas & Aumentado-Armstrong, 2021; Nikishin et al., 2022; Ni et al., 2024) as a challenging benchmark by varying noise dimensions. However, our findings in DMC show that this protocol does not present sufficient difficulty for the best-performing method. In pixel-based domains, IID Gaussian noise is less common but has been studied in Zhang et al. (2018) on Atari games. Unlike our findings, they report that Atari with natural video backgrounds is more challenging than with Gaussian noises, possibly due to domain differences (Atari vs. DMC). Similar to our IID Gaussian with random projection setting, Voelcker et al. (2024) projects observations using a random binary matrix in discrete domains.

In pixel-based domains, beyond the commonly used grayscale video distractions (Zhang et al., 2020; Kemertas & Aumentado-Armstrong, 2021; Zang et al., 2022; Chen & Pan, 2022), several other works have explored injecting distracting video backgrounds. In DeepMind Control Remastered (Grigsby & Qi, 2020), environments are initialized with a visual seed that randomly selects graphical elements – such as floor textures, backgrounds, body colors, and camera/lighting settings – to drastically alter the appearance of image renderings while leaving the transition dynamics unchanged, allowing millions of distinct visualizations of the same state sequence. The Distracting Control Suite (Stone et al., 2021) provides a benchmark by introducing challenging visual distractions including similar graphical variations, resulting in a spectrum of difficulty levels. The DM-Control Generalization Benchmark (DMC-GB) (Hansen & Wang, 2021) provides a framework for evaluating agents' generalization ability in pixel-based DMC tasks with distractions from random colors and video backgrounds. Unlike our setting, it trains agents in a fixed, clean background and evaluates them in a distracted environment, emphasizing the effectiveness of image-augmentation-based methods. RL-ViGen (Yuan et al., 2023) creates a benchmark including domains across indoor navigation, autonomous driving, and robotic manipulation, and various generalization types including visual appearances, camera views, lighting conditions, scene structures, and cross embodiments.

Feature Normalization in RL. Recent work has demonstrated the benefits of layer normalization (LayerNorm) (Ba et al., 2016) in deep RL, especially in state-based domains (Hiraoka et al., 2021; Smith et al., 2022; Nauman et al., 2024). Our findings also highlight LayerNorm's critical role in state-based tasks but with key differences. Unlike most prior work, we apply LayerNorm only to the encoder's output layer, not every dense layer, as in Fujimoto et al. (2023). Additionally, we demonstrate the combined impact of LayerNorm and self-prediction on distracting tasks, rather than in purely model-free RL on clean tasks. In pixel-based domains, most prior metric learning methods incorporate LayerNorm in the encoder following SAC-AE (Yarats et al., 2021b) but do not evaluate its contribution in ablation studies.

Beyond LayerNorm, L_2 normalization is also widely used in RL (Hussing et al., 2024). SimSR (Zang et al., 2022) demonstrates its effectiveness for its own method in pixel-based domains. SPR (Schwarzer et al., 2020) integrates L_2 normalization into the self-prediction loss, effectively transforming MSE into cosine loss. However, unlike our implementation, SPR does not normalize representations for the actor or critic loss.

Finally, various other normalization techniques (Bhatt et al., 2019; Fujimoto et al., 2023; Li et al., 2023; Hansen et al., 2023) have been explored in RL, focusing on sample efficiency and generalization. Our work complements these efforts by analyzing the role of LayerNorm in representation and metric learning on distracting tasks.

Beyond Behavioral Metrics. The idea of learning metrics that represent state discrepancy and embedding them into the representation space (Eq. 2) can be extended to a broader context beyond behavioral metrics that rely heavily on the reward signal. As prominent examples, Agarwal et al. (2021a) utilize a state distance on optimal policies to shape the representation. Wang et al. (2023)

introduce an inverse dynamics bisimulation metric that incorporates the discrepancy in predicted inverse dynamics into the PBSM formulation. Similarly, Rudolph et al. (2024) propose action-bisimulation metric that captures the equivalence of states in terms of action controllability only, where a small distance indicates that the states share similar inverse dynamics on the representation space. Myers et al. (2024) highlight the link between metrics and goal-conditioned RL by introducing a reward-free temporal distance between a state and a goal.

B Notation

Table 3 presents the abbreviations used frequently in our work and their full names.

Table 4, Table 5, and Table 6 show the glossary used in this paper.

Table 3: Abbreviations and their full names.

Abbreviation	Full Name
RP	Reward Prediction
ZP	Self-prediction
BMDP	Block Markov Decision Process (Du et al., 2019)
EX-BMDP	Exogenous BMDP (Efroni et al., 2021)
BSM	Bisimulation Metric (Ferns et al., 2004; 2011)
PBSM	Policy-dependent Bisimulation Metric (Castro, 2020)
SAC	Soft Actor-Critic (Haarnoja et al., 2018)
SAC-AE	Soft Actor-Critic + Autoencoder (Yarats et al., 2021b)
DBC	Deep Bisimulation Control (Zhang et al., 2020)
DBC-normed	DBC with max normalization (Kemertas & Aumentado-Armstrong, 2021)
MICo	Matching under Independent Couplings (Castro et al., 2021)
SimSR	Simple Distance-based State Representation (Zang et al., 2022)
RAP	Reducing Approximation Gap (Chen & Pan, 2022)
DF	Denoising Factor
IID	Independent and Identically Distributed
ID Generalization	In-distribution Generalization
OOD Generalization	Out-of-distribution Generalization
DMC	DeepMind Control Suite (Tassa et al., 2018; Tunyasuvunakool et al., 2020)

C Background on Metrics and Metric Learning

C.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[11]**.** 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 $d(x,y) = 0 \iff x = y$.

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

Table 4: 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

Table 5: Glossary of notations in RL agents.

Symbol	Description
$\psi \in \Psi$	Agent's representation
$\phi : \mathcal{X} \to \Psi$	Agent's encoder
$\tilde{\phi} : \mathcal{X} \to \Psi$	(Isolated) Metric encoder
$\pi_\theta(a \mid \psi)$	(Latent) Actor
$Q_\omega(\psi, a)$	(Latent) Critic
$R_\kappa(\psi, a)$	(Latent) Reward model
$P_\nu(\psi' \mid \psi, a)$	(Latent) Transition model
$\mathcal{P}_\phi(\psi' \mid x, a)$	Grounded-to-latent transition

Table 6: Glossary of notations in metrics.

Symbol	Description
$d_\mathcal{X} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$	Target metric
$d_\Psi : \Psi \times \Psi \to \mathbb{R}$	Representational metric
$d_R : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$	Reward distance
$d_T(d)(\cdot, \cdot)$	Transition distance
$x, x_+, x_- \in \mathcal{X}$	Anchor, positive, negative
$\mathrm{DF}_d^\pi(\phi)$	Denoising factor

C.2 Definitions of Various Behavioral Metrics

In this section, we discuss the behavioral metrics for an EX-BMDP (Sec. 2.1) that serve as candidates for $d_\mathcal{X}$ in Sec. 3. From the observation space $\mathcal{X}$, the grounded transition function is defined as $\mathcal{P}(x' \mid x, a) = \sum_{z' \in \mathcal{Z}} q(x' \mid z') p(z' \mid q^{-1}(x), a)$ and the grounded reward function as $\mathcal{R}(x, a) = R(\phi^*(x), a)$. Let $x_1, x_2 \in \mathcal{X}$ 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. 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 pseudometric $d^\sim : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, called the bisimulation metric (BSM)*[12]*, defined as:*

$$d^\sim(x_1, x_2) := \max_{a \in \mathcal{A}} \left(c_R |\mathcal{R}(x_1, a) - \mathcal{R}(x_2, a)| + c_T \mathcal{W}_1(d^\sim)(\mathcal{P}(\cdot \mid x_1, a), \mathcal{P}(\cdot \mid x_2, a)) \right), \quad (10)$$

where 1-Wasserstein (Kantorovich) distance $\mathcal{W}_1(d^\sim)(P, Q) = \inf_{\delta \in \mathcal{T}(P,Q)} \mathbb{E}_{(x_1', x_2') \sim \delta}[d^\sim(x_1', x_2')]$ with $\mathcal{T}(P, Q)$ the coupling space for P and Q. Here, c_R and c_T are coefficients for short-term and long-term behavioral differences, which are commonly set to $c_R = 1$ and $c_T = \gamma$, where γ is the MDP's discount factor.

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:*

$$d^\pi(x_1, x_2) := c_R |\mathcal{R}^\pi(x_1) - \mathcal{R}^\pi(x_2)| + c_T \mathcal{W}_1(d^\pi)(\mathcal{P}^\pi(\cdot \mid x_1), \mathcal{P}^\pi(\cdot \mid x_2)), \quad (11)$$

[12] 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 the existence of such a fixed-point.

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 closed-form solution under such assumption (Sec. C.3). To further circumvent the costly computation of the 1-Wasserstein distance, Castro et al. (2021) propose MICo distance which uses the independent coupling rather than all coupling of the distributions in the 1-Wasserstein distance, which enables its computation using samples.

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:*

$$u^\pi(x_1, x_2) := c_R|\mathcal{R}^\pi(x_1) - \mathcal{R}^\pi(x_2)| + c_T\mathbb{E}_{x_1'\sim\mathcal{P}^\pi(\cdot|x_1), x_2'\sim\mathcal{P}^\pi(\cdot|x_2)}[u^\pi(x_1', x_2')]. \tag{12}$$

The independent coupling term (second term on the RHS) may yield a non-zero distance between states that share identical immediate rewards and transition dynamics for all actions. Consequently, the MICo distance can be characterized as a diffuse metric (see Sec. C.1). 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:*

$$u^\pi(x_1, x_2) \;:=\; c_R\big|\mathcal{R}^\pi(x_1) - \mathcal{R}^\pi(x_2)\big| \;+\; c_T\,\mathbb{E}_{x_1'\sim\widehat{\mathcal{P}}^\pi(\cdot|x_1),\, x_2'\sim\widehat{\mathcal{P}}^\pi(\cdot|x_2)}\big[u^\pi(x_1',\, x_2')\big], \tag{13}$$

where $\widehat{\mathcal{P}}^\pi$ is an approximated transition dynamics model. Specifically in SimSR, through isometric embedding (Eq. 2), $u^\pi(x_1, x_2) = d_\mathcal{X}(x_1, x_2) = d_\Psi(\phi(x_1), \phi(x_2)) = 1 - \cos\big(\phi(x_1), \phi(x_2)\big)$, which is the cosine distance (normalized dot product distance).

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:*

$$u^\pi(x_1, x_2) \;:=\; c_R\big|\mathcal{R}^\pi(x_1) - \mathcal{R}^\pi(x_2)\big| \;+\; c_T\,\mathbb{E}_{a_1\sim\pi,\, a_2\sim\pi}\Big[u^\pi\Big(\mathbb{E}_{x_1'\sim\hat{\mathcal{P}}(\cdot|x_1,a_1)}[x_1'],\, \mathbb{E}_{x_2'\sim\hat{\mathcal{P}}(\cdot|x_2,a_2)}[x_2']\Big)\Big]. \tag{14}$$

C.3 Approximating the Behavioral Metrics

Behavioral metrics are approximated and isometrically embedded into the representation space via an auxiliary loss in recent works. In this section, we extend the explanation of design choices used to approximate the metrics in their implementations presented in Table 1.

In DBC (Zhang et al., 2020) and DBC-normed (Kemertas & Aumentado-Armstrong, 2021), to approximate PBSM (Def. 6), the metric loss is defined in the following form:

$$J_M(\phi) = \Bigg(\underbrace{\|\phi(x_1) - \phi(x_2)\|_1}_{=d_\Psi(\phi(x_1),\phi(x_2))} - \underbrace{\Big(|r_1 - r_2| \;+\; \gamma\,\mathcal{W}_2(\|\cdot\|_1)\Big(\hat{\mathcal{P}}\Big(\psi' \mid \bar{\phi}(x_1), a_1\Big), \hat{\mathcal{P}}\Big(\psi' \mid \bar{\phi}(x_2), a_2\Big)\Big)\Big)}_{\approx d_R(x_1,x_2)+d_T(d_\Psi)(\mathcal{P}_\phi(\psi'|x_1),\mathcal{P}_\phi(\psi'|x_2))=d_\mathcal{X}(x_1,x_2)}\Bigg)^2. \tag{15}$$

where $(x_1, x_2, a_1, a_2, r_1, r_2)$ is sampled from a replay buffer, $\bar{\phi}$ denotes the encoder ϕ with gradient detached, the transition model $\hat{\mathcal{P}}$ outputs a *factorized* Gaussian distribution, with mean $\mu_{\hat{\mathcal{P}}}(\bar{\phi}(x), a)$ and covariance $\sigma_{\hat{\mathcal{P}}}(\bar{\phi}(x), a)$. For brevity, we denote $\mu_1 = \mu_{\hat{\mathcal{P}}}(\bar{\phi}(x_1), a_1)$ and $\sigma_1 = \sigma_{\hat{\mathcal{P}}}(\bar{\phi}(x_1), a_1)$. $\mathcal{W}_2$ is the 2-Wasserstein distance, which serves as a surrogate for the 1-Wasserstein distance in Def. 6 and admits a convenient closed-form solution when comparing two Gaussian distributions:

$$\mathcal{W}_2(\|\cdot\|_2)\Big(\mathcal{N}(\mu_1, \Sigma_1), \mathcal{N}(\mu_2, \Sigma_2)\Big) = \sqrt{\|\mu_1 - \mu_2\|_2^2 \;+\; \big\|\Sigma_1^{\frac{1}{2}} - \Sigma_2^{\frac{1}{2}}\big\|_F^2}\,. \tag{16}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\Sigma_1, \Sigma_2 \in \mathbb{R}^{k\times k}$ are the covariance matrices. In the special case where both Gaussians are factorized (i.e., $\Sigma_i = \mathrm{diag}(\sigma_i^2)$), the Frobenius norm simplifies to the Euclidean norm over the standard deviations, yielding:

$$\mathcal{W}_2(\|\cdot\|_2)\Big(\mathcal{N}(\mu_1, \mathrm{diag}(\sigma_1^2)), \mathcal{N}(\mu_2, \mathrm{diag}(\sigma_2^2))\Big) = \sqrt{\|\mu_1-\mu_2\|_2^2 \;+\; \|\sigma_1-\sigma_2\|_2^2}\,. \tag{17}$$

In their implementation, several major modifications are applied to Eq. 15. First of all, DBC and DBC-normed use a *scaled Huber distance* instead of the L_1 distance as the approximant of representation distance. We first define the Huber distance[13] for two vectors $x, y \in \mathbb{R}^k$ as:

$$d_{\mathrm{Huber}}(x,y) = \sum_{i=1}^{k} \begin{cases} \frac{1}{2}, (x_i-y_i)^2, & \text{if } |x_i-y_i|<1, \\ |x_i-y_i| - \frac{1}{2}, & \text{otherwise.} \end{cases} \tag{18}$$

The scaled Huber distance can be formulated as:

$$d_\Psi\big(\phi(x_1),\, \phi(x_2)\big) = \frac{1}{k} d_{\mathrm{Huber}}\big(\phi(x_1), \phi(x_2)\big). \tag{19}$$

As for approximating reward distance d_R, instead of using the absolute difference,

$$\hat{d_R}(r_1, r_2) = |r_1 - r_2|, \tag{20}$$

DBC and DBC-normed apply Huber distance:

$$\hat{d}_R(r_1, r_2) = d_{\mathrm{Huber}}\big((r_1), (r_2)\big). \tag{21}$$

When approximating the transition distance d_T, DBC use the following form that slightly differs from Eq. 16 to compare two Gaussian distributions:

$$\hat{d}_T\big((\mu_1,\sigma_1),\, (\mu_2,\sigma_2)\big) = \frac{1}{k}\sum_{i=1}^{k} \sqrt{(\mu_{1,i}-\mu_{2,i})^2 \;+\; (\sigma_{1,i}-\sigma_{2,i})^2}\,. \tag{22}$$

DBC-normed, instead, utilize this form:

$$\hat{d}_T\big((\mu_1,\sigma_1),\, (\mu_2,\sigma_2)\big) = \frac{1}{k}\big(d_{\mathrm{Huber}}(\mu_1,\mu_2) \;+\; d_{\mathrm{Huber}}(\sigma_1,\sigma_2)\big). \tag{23}$$

MICo (Castro et al., 2021) shares a similar metric loss structure with DBC:

$$J_M(\phi) = \Big(\underbrace{U_\phi(x_1,x_2)}_{=d_\Psi(\phi(x_1),\phi(x_2))} - \underbrace{\big(|r_1-r_2| + \gamma\, U_{\bar{\phi}}(x_1', x_2')\big)}_{\approx d_{\mathcal{X}}(x_1,x_2)} \Big)^2, \tag{24}$$

where $(x_1, x_2, r_1, r_2, x_1', x_2')$ is sampled from a replay buffer, $U_\phi(x_1, x_2)$ is the representation distance, parameterized as:

$$d_\Psi(\phi(x_1),\phi(x_2)) = U_\phi(x_1,x_2) \;:=\; \frac{\|\phi(x_1)\|_2^2 + \|\phi(x_2)\|_2^2}{2} \;+\; \beta\,\theta\big(\phi(x_1),\phi(x_2)\big), \tag{25}$$

where θ represents an angular distance function defined in Appendix Sec. C.2 in Castro et al. (2021). In the implementation of MICo, the Huber loss (Huber, 1992) is used instead of MSE in Eq. 24.

SimSR (Zang et al., 2022) differs from MICo in its metric loss along two main dimensions. First, the parameterization of U_ϕ:

$$d_\Psi(\phi(x_1),\phi(x_2)) = U_\phi(x_1,x_2) := 1 - \cos\big(\phi(x_1),\phi(x_2)\big). \tag{26}$$

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

Second, the next latents $\phi(x_1'), \phi(x_2')$ are sampled from a transition model rather than encoded from next observations in the replay buffer.

RAP (Chen & Pan, 2022) reduces the gap of d_R and $\hat{d}_R$ in the prior work by introducing a better approximant of d_R, motivated by the following derivation of d_R:

$$d_R(x_1, x_2) = \left|\mathcal{R}^\pi(x_1) - \mathcal{R}^\pi(x_2)\right| = \sqrt{\mathbb{E}_{a_1\sim\pi,\, a_2\sim\pi}\left[|\mathcal{R}(x_1, a_1) - \mathcal{R}(x_2, a_j)|^2\right] - \mathrm{Var}[r_{x_1}] - \mathrm{Var}[r_{x_2}]}, \tag{27}$$

where r_x is a random variable defined by $p(r_x = \mathcal{R}(x, a)) = \pi(a \mid x)$, and $\mathrm{Var}[r_x]$ is the variance of r_x. To approximate $\mathrm{Var}[r_x]$, RAP introduces an observation-reward model[14] that maps the current observation to the mean and variance of the expected reward. The estimated $\hat{d}_R$ is then computed by substituting $\mathrm{Var}[r_x]$ with the model-predicted variance, and approximating the expected reward difference using sampled reward pairs from a replay buffer. For $\hat{d}_T$, RAP adopts the same computation procedure as DBC.

C.4 Normalization Techniques

Several related normalization techniques and their connections are introduced in this section. We also justify our study design by incorporating LayerNorm in candidate methods rather than the L_2 normalization employed in SimSR (Zang et al., 2022).

Max Normalization. DBC-normed (Kemertas & Aumentado-Armstrong, 2021) derives an upper bound of L_p-norm of the representation, and imposes this bound to the representation space. Specifically, assuming the boundedness of the reward, the target metric $d_\mathcal{X}$ (PBSM in DBC-normed) and the representational metric d_Ψ (through isometric embedding (Eq. 2)) can be upper bounded by a constant C:

$$d_\Psi(\phi(x_1), \phi(x_2)) = d_\mathcal{X}(x_1, x_2) \leq \frac{c_R}{1 - c_T}(\max_{x,a} \mathcal{R}(x, a) - \min_{x,a} \mathcal{R}(x, a)) := C. \tag{28}$$

For example, in our hyperparameter setting in DMC, the constant C can be $100, 200$, or 400, depending on the action repeat. Such a bound can also be naturally generalized to MICo, SimSR, and RAP distance. Specifically, when d_Ψ is the L_p-distance, if the L_p-norm of $\psi = \phi(x)$ is upper bounded as:

$$\|\psi\|_p \leq \frac{C}{2}, \tag{29}$$

then Eq. 28 can be satisfied.

A "max normalization" can then be imposed on ψ to constrain the approximated metrics within a reasonable numerical range, thereby improving metric estimation:

$$\mathrm{MaxNorm}(\psi) := \begin{cases} \psi, & \text{if } \|\psi\|_p < \frac{C}{2}, \\ \frac{C}{2}\frac{\psi}{\|\psi\|_p}, & \text{otherwise.} \end{cases} \tag{30}$$

L_2 **Normalization.** The L_2 normalization of ψ enforces $\|\psi\|_2 = 1$, which is defined as:

$$\mathrm{L2Norm}(\psi) = \frac{\psi}{\|\psi\|_2}. \tag{31}$$

Remark 1 (**Caveat of applying L_2 normalization to metric learning**). Note the boundedness of L_p distance for L_2-normalized vectors $\phi(x_1), \phi(x_2) \in \mathbb{R}^k$ are given as follows by Hölder's inequality:

$$\|\phi(x_1) - \phi(x_2)\|_p \leq 2\|\phi(x_1)\|_p \leq \begin{cases} 2\,k^{\frac{1}{p}-\frac{1}{2}}, & \text{if } 1 \leq p \leq 2, \\ 2, & \text{if } p \geq 2. \end{cases} \tag{32}$$

[14] Note that this model introduces additional gradients that influence the encoder's representation learning.

Note that if the L_p-distance is used as d_Ψ, directly applying L_2 normalization on ψ can lead to the **misspecification of the metrics**. In other words, the numerical range of d_Ψ may not be *sufficiently expressive* to capture the ground truth metrics, i.e., the target metric space $(\mathcal{X}, d_\mathcal{X})$ cannot always be isometrically embedded into the Euclidean unit sphere $(\Psi_{\text{unit}}, \|\cdot\|_p)$. Consider a counterexample when $p = 2$. Suppose the existence of a pair of x_1, x_2 that $c_R|\mathcal{R}^\pi(x_1) - \mathcal{R}^\pi(x_2)| > 2$, and $d_\mathcal{X}(x_1, x_2) > 2$. But $d_\Psi(\phi(x_1), \phi(x_2)) = \|\phi(x_1) - \phi(x_2)\|_p \leq 2$. Thus, in this case, there is no such ϕ that $\|\phi(x_1)\|_2 = \|\phi(x_2)\|_2 = 1$ and $d_\mathcal{X}(x_1, x_2) = d_\Psi(\phi(x_1), \phi(x_2))$.

As another counterexample, in SimSR (Zang et al., 2022), L_2 normalization is imposed on the representation space, i.e., $\|\phi(x_1)\|_2 = \|\phi(x_2)\|_2 = 1$. Under such condition, one can show the equivalence of the cosine distance and the squared L_2 distance:

$$\begin{aligned} d_\Psi(\phi(x_1), \phi(x_2)) &= 1 - \cos\big(\phi(x_1), \phi(x_2)\big) \\ &= 1 - \langle \phi(x_1), \phi(x_2) \rangle \\ &= \frac{1}{2}\Big(\|\phi(x_1)\|_2^2 + \|\phi(x_2)\|_2^2 - 2\langle \phi(x_1), \phi(x_2) \rangle\Big) \\ &= \frac{1}{2}\|\phi(x_1) - \phi(x_2)\|_2^2 \in [0, 2]. \end{aligned} \tag{33}$$

Similarly, in such a case, a ϕ that satisfies $\forall x_1, x_2 \in \mathcal{X}$, $\|\phi(x_1)\|_2 = \|\phi(x_2)\|_2 = 1$ and $d_\mathcal{X}(x_1, x_2) = d_\Psi(\phi(x_1), \phi(x_2))$ does not always exist. □

Layer Normalization. LayerNorm (Ba et al., 2016) can be formalized as follows:

$$\text{LayerNorm}(\psi) = \alpha \odot \frac{\psi - \mu(\psi)}{\sqrt{\sigma^2(\psi) + \epsilon}} + \beta, \tag{34}$$

where $\alpha, \beta \in \mathbb{R}^k$ are learnable parameters, $\epsilon > 0$ is a small constant for numerical stability, and $\odot$ denotes element-wise multiplication. μ and σ^2 are the mean and variance:

$$\mu(\psi) = \frac{1}{k}\sum_{i=1}^{k} \psi_i, \qquad \sigma^2(\psi) = \frac{1}{k}\sum_{i=1}^{k} (\psi_i - \mu(\psi))^2. \tag{35}$$

Note that, if we assume $\alpha = \alpha_0 \mathbf{1}_k \in \mathbb{R}^k$ where α_0 is a constant and $\beta \approx \mathbf{0}$ (as it is initialized to 0 in many implementations and remains small early in training), we have:

$$\begin{aligned} \big\|\text{LayerNorm}(\psi)\big\|_2^2 &\approx \sum_{i=1}^{k} \alpha_i^2 \cdot \frac{(\psi_i - \mu(\psi))^2}{\sigma^2(\psi) + \epsilon} \\ &= \alpha_0^2 \cdot \frac{\sum_{i=1}^{k} (\psi_i - \mu(\psi))^2}{\sigma^2(\psi)} \\ &= \alpha_0^2 \cdot \frac{k\sigma^2(\psi)}{\sigma^2(\psi)} \quad \text{(from variance definition)} \\ &= \alpha_0^2 k. \end{aligned} \tag{36}$$

Then the L_2 norm of the representation after LayerNorm satisfies:

$$\big\|\text{LayerNorm}(\psi)\big\|_2 \approx \alpha_0 \sqrt{k}. \tag{37}$$

For layer-normalized $\phi(x_1)$ and $\phi(x_2)$, the upper bound of $d_\Psi(\phi(x_1), \phi(x_2))$ depends on both α and β, providing flexibility in expressing the target metrics $d_\mathcal{X}$. As a result, we conduct case studies in Sec. 5.2 on methods with LayerNorm rather than L_2 normalization to ensure full expressivity of the target metrics.

D Proof

D.1 Proof of Transition Distance Preservation under Isometric Embedding

Lemma 1 (Transition distance preservation (Eq. 5))**.** *Let $\phi : \mathcal{X} \to \Psi$ be an isometric embedding with $d_{\mathcal{X}}(x_1, x_2) = d_{\Psi}(\phi(x_1), \phi(x_2))$ for any $x_1, x_2 \in \mathcal{X}$ (Eq. 2). Then,*

$$d_T(d_{\mathcal{X}})(\mathcal{P}(x' \mid x_1), \mathcal{P}(x' \mid x_2)) = d_T(d_{\Psi})(\mathcal{P}_\phi(\psi' \mid x_1), \mathcal{P}_\phi(\psi' \mid x_2)). \tag{38}$$

Proof. We prove it by considering two common forms of d_T: Wasserstein distance (Ferns et al., 2004; 2011; Zhang et al., 2020) and sampling-based distance (Castro et al., 2021; Zang et al., 2022).

Case 1: d_T is Wasserstein distance. For convenience, we take 1-Wasserstein distance as example, and the proof naturally extends to p-Wasserstein. Let $\mathcal{T}(P, Q)$ be the coupling space for P and Q,

$$\begin{aligned}
&\mathcal{W}_1(d_{\mathcal{X}})(\mathcal{P}(x' \mid x_1), \mathcal{P}(x' \mid x_2)) = \inf_{\mu \in \mathcal{T}(\mathcal{P}(x'|x_1), \mathcal{P}(x'|x_2))} \sum_{x_1' \in \mathcal{X}, x_2' \in \mathcal{X}} d_{\mathcal{X}}(x_1', x_2') \mu(x_1', x_2') \\
&= \inf_{\mu \in \mathcal{T}(\mathcal{P}(x'|x_1), \mathcal{P}(x'|x_2))} \sum_{x_1' \in \mathcal{X}, x_2' \in \mathcal{X}} d_{\Psi}(\phi(x_1'), \phi(x_2')) \mu(x_1', x_2') \quad \text{(isometric embedding)} \\
&= \inf_{\mu \in \mathcal{T}(\mathcal{P}(x'|x_1), \mathcal{P}(x'|x_2))} \sum_{\psi_1' \in \Psi, \psi_2' \in \Psi} d_{\Psi}(\psi_1', \psi_2') \sum_{x_1', x_2' : \phi(x_1') = \psi_1', \phi(x_2') = \psi_2'} \mu(x_1', x_2') \\
&= \inf_{\nu \in \mathcal{T}(\mathcal{P}_\phi(\psi'|x_1), \mathcal{P}_\phi(\psi'|x_2))} \sum_{\psi_1' \in \Psi, \psi_2' \in \Psi} d_{\Psi}(\psi_1', \psi_2') \nu(\psi_1', \psi_2') \\
&= \mathcal{W}_1(d_{\Psi})(\mathcal{P}_\phi(\psi' \mid x_1), \mathcal{P}_\phi(\psi' \mid x_2)).
\end{aligned} \tag{39}$$

Case 2: d_T as sampling-based distance.

$$\begin{aligned}
&\mathbb{E}_{x_1' \sim \mathcal{P}(\cdot|x_1), x_2' \sim \mathcal{P}(\cdot|x_2)}[d_{\mathcal{X}}(x_1', x_2')] = \sum_{x_1' \in \mathcal{X}, x_2' \in \mathcal{X}} d_{\mathcal{X}}(x_1', x_2') \mathcal{P}(x_1' \mid x_1) \mathcal{P}(x_2' \mid x_2) \\
&= \sum_{x_1' \in \mathcal{X}, x_2' \in \mathcal{X}} d_{\Psi}(\phi(x_1'), \phi(x_2')) \mathcal{P}(x_1' \mid x_1) \mathcal{P}(x_2' \mid x_2) \quad \text{(isometric embedding)} \\
&= \sum_{\psi_1' \in \Psi, \psi_2' \in \Psi} d_{\Psi}(\psi_1', \psi_2') \sum_{x_1', x_2' : \phi(x_1') = \psi_1', \phi(x_2') = \psi_2'} \mathcal{P}(x_1' \mid x_1) \mathcal{P}(x_2' \mid x_2) \\
&= \sum_{\psi_1' \in \Psi, \psi_2' \in \Psi} d_{\Psi}(\psi_1', \psi_2') \left(\sum_{x_1' : \phi(x_1') = \psi_1'} \mathcal{P}(x_1' \mid x_1) \right) \left(\sum_{x_2' : \phi(x_2') = \psi_2'} \mathcal{P}(x_2' \mid x_2) \right) \\
&= \mathbb{E}_{\psi_1' \sim \mathcal{P}_\phi(\cdot|x_1), \psi_2' \sim \mathcal{P}_\phi(\cdot|x_2)}[d_{\Psi}(\psi_1', \psi_2')].
\end{aligned} \tag{40}$$

□

D.2 Proof of Denoising Property of Bisimulation Metrics

Proposition 1 (**Denoising property of BSM**)**.** *For any $x, x_+ \in \mathcal{X}$ of an EX-BMDP (Sec. 2.1) with $\phi^*(x) = \phi^*(x_+)$, the bisimulation metric (Def. 5) is zero: $d^\sim(x, x_+) = 0$.*

Proof. Let $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be any pseudometric. We define the bisimulation operator $\mathcal{F}$ on d: $\forall x_1, x_2 \in \mathcal{X}$,

$$(\mathcal{F}(d))(x_1, x_2) := \max_{a \in \mathcal{A}} \left(c_R |\mathcal{R}(x_1, a) - \mathcal{R}(x_2, a)| + c_T \mathcal{W}_1(d)(\mathcal{P}(x' \mid x_1, a), \mathcal{P}(x' \mid x_2, a)) \right). \tag{41}$$

We initialize $d^{(0)}(x_1, x_2) = 0, \forall x_1, x_2$ and then apply the operator: $d^{(n+1)} = \mathcal{F}(d^{(n)}), \forall n \in \mathbb{N}$. Iteratively, the metric will converge to the unique fixed point $d^\sim$ (Ferns et al., 2004; 2011): $\lim_{n \to \infty} d^{(n)} = d^\sim$.

We show by induction that $d^{(n)}(x, x_+) = 0$ whenever $\phi^*(x) = \phi^*(x_+)$, and hence $d^{\sim}(x, x_+) = 0$ in the limit. The base case $n = 0$ is immediate, because $d^{(0)} \equiv 0$. Assume for some $n \in \mathbb{N}$, $d^{(n)}(x, x_+) = 0$ whenever $\phi^*(x) = \phi^*(x_+)$. Consider the $n+1$-case:

$$d^{(n+1)}(x, x_+) = \max_{a \in \mathcal{A}} \Big(c_R |\mathcal{R}(x, a) - \mathcal{R}(x_+, a)| + c_T \mathcal{W}_1(d^{(n)})(\mathcal{P}(x' \mid x, a), \mathcal{P}(x' \mid x_+, a)) \Big). \tag{42}$$

Recall in an EX-BMDP, $\mathcal{R}(x, a)$ only depends on $s = \phi^*(x)$ and a. Since $\phi^*(x) = \phi^*(x_+)$, the reward difference term is zero. The remainder of the expression is governed by the transition distance term:

$$\mathcal{W}_1(d^{(n)})(\mathcal{P}(x' \mid x, a), \mathcal{P}(x' \mid x_+, a)) = \inf_{\mu \in \mathcal{T}(\mathcal{P}(\cdot|x,a), \mathcal{P}(\cdot|x_+,a))} \sum_{x', x'_+ \in \mathcal{X}} d^{(n)}(x', x'_+) \mu(x', x'_+), \tag{43}$$

where $\mathcal{T}(P, Q)$ is the coupling space for P and Q. Consider the coupling that pairs next-state samples by forcing them to share the same task-relevant state. Let ξ, ξ_+ be the task-irrelevant noise underlying x, x_+ (i.e., $[\phi^*(x), \xi] = q^{-1}(x)$). Since $\phi^*(x) = \phi^*(x_+)$, the coupling μ is defined by

$$\mu(x', x'_+) = \sum_{s' \in \mathcal{S}, \xi' \in \Xi, \xi'_+ \in \Xi} p(s' \mid \phi^*(x), a) p(\xi' \mid \xi) p(\xi'_+ \mid \xi_+) q(x' \mid s', \xi') q(x'_+ \mid s', \xi'_+). \tag{44}$$

In this construction, any sample (x', x'_+) shares the same task-relevant state s'. By the inductive hypothesis, $d^{(n)}(x', x'_+) = 0$. Thus,

$$\sum_{x', x'_+ \in \mathcal{X}} d^{(n)}(x', x'_+) \mu(x', x'_+) = \sum_{x', x'_+ \in \mathcal{X}} 0 = 0. \tag{45}$$

Therefore, $\mathcal{W}_1(d^{(n)})(\mathcal{P}(x' \mid x, a), \mathcal{P}(x' \mid x_+, a)) = 0$ and $d^{(n+1)}(x, x_+) = 0$. □

Proposition 2 (**Denoising property of PBSM with an exo-free policy**). *Define an exo-free policy (Islam et al., 2022) $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ that is independent of noise, i.e., $\pi(a \mid x) = \pi(a \mid x_+), \forall a \in \mathcal{A}$ whenever $\phi^*(x) = \phi^*(x_+)$. For any $x, x_+ \in \mathcal{X}$ of an EX-BMDP with $\phi^*(x) = \phi^*(x_+)$, the policy-dependent bisimulation metric (Def. 6) with an exo-free policy π is zero: $d^\pi(x, x_+) = 0$.*

Proof. This is a proof sketch, adapting the bisimulation metric (BSM) proof above to PBSM.

By induction, consider the $n+1$-case:

$$d^{(n+1)}(x, x_+) = \Big(c_R |\mathcal{R}^\pi(x) - \mathcal{R}^\pi(x_+)| + c_T \mathcal{W}_1(d^{(n)})(\mathcal{P}^\pi(x' \mid x), \mathcal{P}^\pi(x' \mid x_+)) \Big). \tag{46}$$

Since $\phi^*(x) = \phi^*(x_+)$ and both reward and policy are exo-free, we have $\mathcal{R}^\pi(x) = \sum_a \pi(a \mid x) \mathcal{R}(x, a) = \sum_a \pi(a \mid x_+) \mathcal{R}(x_+, a) = \mathcal{R}^\pi(x_+)$, i.e., the reward difference term is zero. Consider the transition difference term, we construct a similar coupling:

$$\mu(x', x'_+) = \sum_{a \in \mathcal{A}, s' \in \mathcal{S}, \xi' \in \Xi, \xi'_+ \in \Xi} \pi(a \mid x) p(s' \mid \phi^*(x), a) p(\xi' \mid \xi) p(\xi'_+ \mid \xi_+) q(x' \mid s', \xi') q(x'_+ \mid s', \xi'_+). \tag{47}$$

In this coupling, any sample (x', x'_+) shares the same task-relevant state s'. Hence $d^{(n+1)}(x, x_+) = 0$ and by the convergence of PBSM operator (Castro, 2020), we have $d^\pi(x, x_+) = 0$. □

Remark 2 (**PBSM may *not* exhibit denoising property at convergence**). In general, an optimal policy is *not necessarily* exo-free. That implies even if a policy converges to an optimal one, denoted as π^*, the corresponding PBSM, d^{π^*}, may still lack the denoising property.

Consider the following counterexample. For $x, x_+ \in \mathcal{X}$ be an anchor-positive pair with the same task-relevant state. By bisimulation relation (Li et al., 2006), they have the same optimal-value

function: $Q^*(x,a) = Q^*(x_+,a), \forall a \in \mathcal{A}$. Now, suppose there exist two distinct optimal actions $a_1, a_2 \in \mathcal{A}$, such that $Q^*(x,a_1) = Q^*(x,a_2) = \max_{a\in\mathcal{A}} Q^*(x,a)$ and $a_1 \neq a_2$. Construct a deterministic optimal policy, π^*, such that $\pi^*(x) = a_1$ and $\pi^*(x_+) = a_2$. In this case, although $Q^*(x,a_1) = Q^*(x_+,a_2)$, the reward difference $|\mathcal{R}^\pi(x) - \mathcal{R}^\pi(x_+)|$ can be still nonzero when $\mathcal{R}(\phi^*(x),a_1) \neq \mathcal{R}(\phi^*(x),a_2)$. Thus, for this optimal policy π^*, $d^{\pi^*}(x,x_+)$ can also be nonzero.

This example indicates that the PBSM under an optimal policy – namely, the fixed point attained by PBSM-based methods (e.g., DBC, DBC-normed) – may not exhibit the denoising property. □

In fact, similarly to Remark 2 illustrating negative results for policy-dependent metrics, we find that model-irrelevance abstraction does not necessarily imply Q^π-irrelevance abstraction, thereby clarifying the scope of the classic result (Li et al., 2006).

Remark 3 (**Model-irrelevance abstraction ϕ_{model} may *not* imply Q^π-irrelevance abstraction ϕ_{Q^π}**). In (Li et al., 2006, Theorem 2), it is established that ϕ_{model} implies ϕ_{Q^π} for *any* policy π. Here, we revisit this statement and demonstrates that it does not extend to *exo-dependent* policies. To see why the classical result fails in this setting, let $x, x_+ \in \mathcal{X}$ be an anchor-positive pair with the same task-relevant state. If ϕ_{model} implies ϕ_{Q^π}, we would have $Q^\pi(x,a) = Q^\pi(x_+,a), \forall a \in \mathcal{A}, \forall \pi$.

However, consider a counterexample similar to Remark 2. Suppose x, x_+ deterministically transits to x', x'_+, respectively under an action a, and the policy π is deterministic. Since $Q^\pi(x,a) = R(x,a) + \gamma Q^\pi(x',\pi(x'))$ and $\mathcal{R}(x,a) = \mathcal{R}(x_+,a)$, the key question becomes $Q^\pi(x',\pi(x')) \stackrel{?}{=} Q^\pi(x'_+,\pi(x'_+))$. Although x', x'_+ are bisimilar, an *exo-dependent* policy may select different actions for these two states, i.e., $\pi(x') \neq \pi(x'_+)$. In a terminal step of a finite-horizon MDP, for example, we could have $Q^\pi(x',\pi(x')) = \mathcal{R}(x',\pi(x')) \neq \mathcal{R}(x'_+,\pi(x'_+)) = Q^\pi(x'_+,\pi(x'_+))$. Therefore, $\phi_{\text{model}} \implies \phi_{Q^\pi}$ does not hold once we allow for exo-dependent policies. □

E Difficulty Levels of the Tasks

Difficulty levels for each task in state-based and pixel-based DMC are provided in Table 7 and Table 8 respectively. These levels are based on the average reward across all compared methods in Table 1. Tasks listed in Table 7 and Table 8 are sorted in ascending order of difficulty. This ordering is also applied to all other per-task performance tables and figures, facilitating a clearer understanding of how different methods perform across tasks of varying difficulty levels.

F Implementation Details

F.1 Hyperparameters

Our hyperparameter settings for the benchmarked agents are based on their open-source codebases and the values reported in their respective papers. Table 9 (above the double rule) shows the general hyperparameter setting adopted by most agents. For the 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). Exceptions to Table 9 are detailed below:

- For MICo (Castro et al., 2021)[15], β in MICo distance parametrization is set to 0.1. In MICo's code, they use a different hidden unit size 1024 rather than 256 in our implementation, and a reward scale of 0.1 rather than 1 in our implementation. We assume that changing the hidden unit size alters the network architecture, and modifying the reward scale effectively changes the environment. Since other algorithms are evaluated under a unified setting, we avoid such modifications to isolate algorithmic differences, which are the primary focus of our study.

[15] https://github.com/google-research/google-research/tree/master/mico

Table 7: **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 8: **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

- For RAP (Chen & Pan, 2022)[16], we use the same hyperparameter setting in their open-source code, where the actor, critic, and encoder learning rates are set to 5×10^{-4}, β in MICo distance

[16] https://github.com/jianda-chen/RAP_distance

parametrization is set to 10^{-6} (which actually almost disables metric learning), RP and ZP loss coefficients are set to 10^{-4}, and the encoder feature dimensionality is set to 100.

Table 9: **Hyperparameter settings** for most *agents* we benchmarked (above the double rule) and for *environmental* configurations (below the double rule).

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 unit size in neural networks	256
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 coefficient λ_{M}	0.5
Metric reward coefficient c_R	1
Metric transition coefficient c_T	0.99
RP loss coefficient λ_{RP}	1
ZP loss coefficient λ_{ZP}	1
Image size	$84 \times 84 \times 3$
Frame stack	3
Paralleled environments	10
Distracting video frames N (per paralleled environment)	1000

F.2 Model Architecture

The general model architectures of all our implemented methods and isolated metric estimation setting are illustrated in Fig. 8, Fig. 9, Fig. 10, and Fig. 11, respectively. For brevity, only the forward pass of the neural network is shown, referring to the computation from inputs to outputs without gradient updates.

F.3 Pixel-based Environmental Setup

In our pixel-based settings, following (Zhang et al., 2020), we use natural videos and images from the Kinetics-400 dataset (Kay et al., 2017) labeled `driving_car` as distracting backgrounds. The dataset is split into 817 training and 90 test video clips, each 10 seconds long, and is publicly available in our codebase (footnote 2).

For each run, a subset of videos of the training set is sampled for use during training, with the total number of frames determined by the hyperparameter "distracting video frame N" in Table 9; that is, the sampled videos must collectively provide at least N frames. In our experiments, we set $N = 1000$, sampling 4 to 5 videos depending on their frame rates. The test set is used for OOD generalization evaluation, ensuring that videos seen during training are excluded from evaluation.

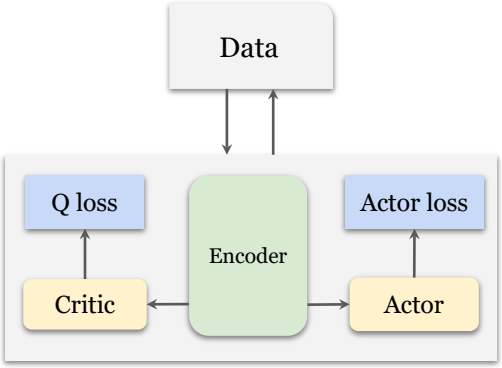

Figure 8: **SAC** architecture used in our experiments.

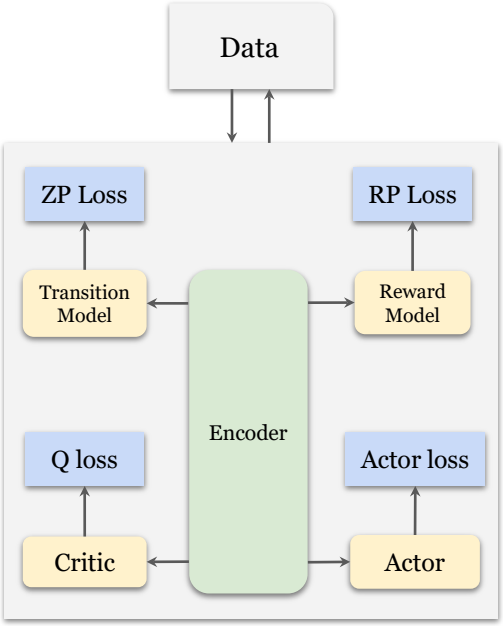

Figure 9: **DeepMDP** architecture used in our experiments.

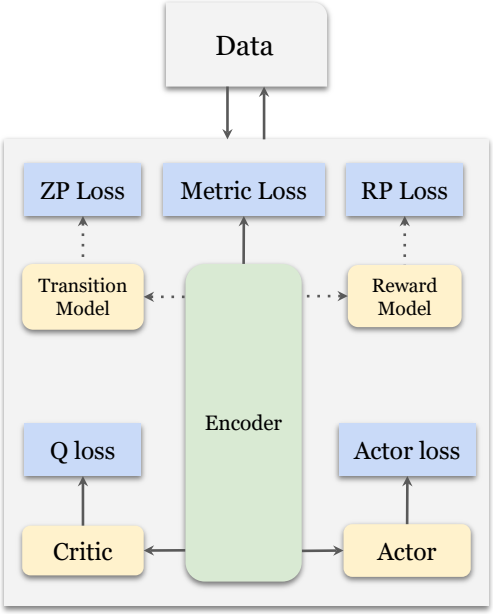

Figure 10: General architecture of **metric learning methods**, summarizing the architecture used in our benchmarked metric learning methods (Table 1). Dotted lines show optional data flows.

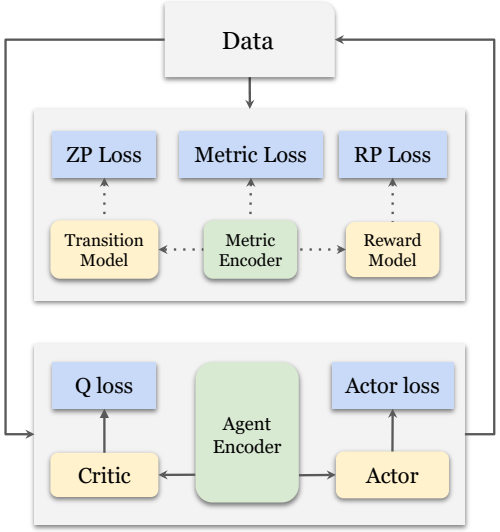

Figure 11: An instance of architecture described in the **isolated metric estimation setting** (Sec. 4.4). Shown is the case where a base agent, SAC (bottom gray module), is used to collect experiences, which are then used to train the isolated metric encoder, as implemented in our experiments in Sec. 5.3. Dotted lines show optional data flows. For example, in Sec. 5.3, isolated metric estimation for DeepMDP enables data flow only to the ZP and RP losses, whereas for MICo, only the metric loss is used.

During OOD evaluation, each parallel environment is assigned a unique set of N distracting frames, which remain fixed within a run. For example, in our experiments with $N = 1000$ and 10 parallel environments, a total of 10000 frames are sampled and fixed during each run.

We elaborate on the instantiation of pixel-based noise settings introduced in Sec. 4.1. In the natural video noise setting, clean background pixels are replaced with those from distracting videos; each parallel environment independently samples an initial starting index ξ_0, from which the video is played sequentially as background. In the natural image noise setting, background pixels are replaced with those from a single randomly sampled frame in the video dataset, which remains fixed across all parallel environments throughout each run.

Additionally, Table 9 (below the double rule) presents the environmental hyperparameters. Table 10 quantifies the distraction in pixel-based distracting DMC tasks by presenting the percentage of noised pixels.

Table 10: Percentage of distracting pixels (the pixels that are task-irrelevant) for 14 pixel-based DMC tasks. These ratios remain consistent across all different pixel-based noise settings introduced in Sec. 4.1.

Task		**Noise Ratio (%)**
cartpole	balance	98.3%
cartpole	balance_sparse	98.3%
walker	stand	92.6%
finger	spin	94.3%
cartpole	swingup	98.3%
ball_in_cup	catch	99.0%
walker	walk	92.6%
point_mass	easy	99.7%
cartpole	swingup_sparse	98.3%
reacher	easy	96.5%
pendulum	swingup	98.9%
cheetah	run	95.4%
walker	run	92.6%
hopper	hop	97.3%

G Additional Experiment Results

Table 11 summarizes the **per-task result figures** presented in this section, each corresponding to an aggregated score reported in the main text. Detailed noise settings are indicated at the top of each figure and table.

Additionally, Table 12 reports model update time comparisons from Sec. 5.1. Fig. 28 presents a case study on six state-based DMC tasks using DBC-normed with LayerNorm and its design variants, as discussed in Sec. 5.2. Fig. 12 and Fig. 30 serve as alternative illustrations of Fig. 2 and Fig. 4, highlighting the agents' sensitivity to noise hyperparameters.

Table 11: Summary of extended experimental results in Appendix. Per-task results corresponding to all aggregated scores shown in the main text are included.

Section	**Description of Settings**	**Main Text Reference**	**Per-task Figures / Tables**
Sec. 5.1	Noise std. sweep: $\sigma \in \{0.2, 1.0, 2.0, 4.0, 8.0\}$ (fixed $m = 32$)	Fig. 2	Table 13 – Table 17 Fig. 13 – Fig. 17
	Noise dim. sweep: $m \in \{2, 16, 32, 64, 128\}$ (fixed $\sigma = 1.0$)	Fig. 2	Table 18 – Table 21 Fig. 18 – Fig. 21
Sec. 5.2	LayerNorm ablation	Table 2	Fig. 29
	IID Gaussian + Random projection noise std. sweep: $\sigma \in \{0.2, 1.0, 2.0, 4.0, 8.0\}$ (fixed $m = 32$)	Fig. 4	Fig. 31 – Fig. 35
Sec. 5.3	SAC (with LayerNorm) reward curves	Fig. 5	Fig. 36
	DF curves on agent encoders co-trained with RL	Fig. 5	Fig. 37
	Pixel-based isolated evaluation setting curves	–	Fig. 38 – Fig. 42
Sec. 5.4	OOD generalization reward curves	Fig. 7	Fig. 43 – Fig. 46
	Reward difference curves	Fig. 6	Fig. 47 – Fig. 48

Table 12: **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 for all methods to Table 9.

	SAC	DeepMDP	DBC	DBC-normed	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

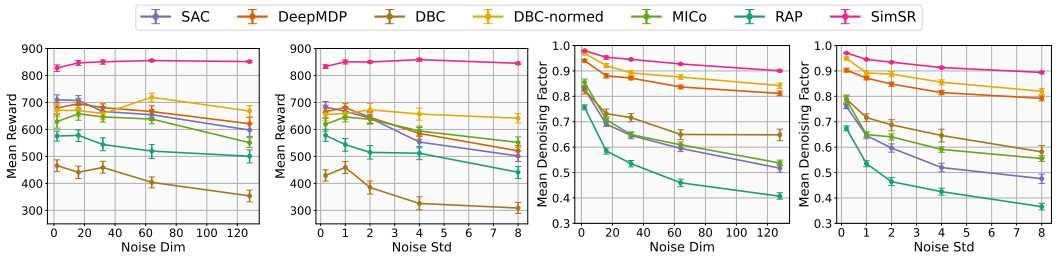

Figure 12: **Benchmarking results: reward (left) and denoising factor (right)** of seven methods to IID Gaussian noise dimension (Noise Dim) and standard deviation (Noise Std). Each point is aggregated by 20 state-based tasks in Table 7.

Table 13: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=0.2, noise dim=32.

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

Table 14: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=32.

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

Table 15: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=2, noise dim=32.

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

Table 16: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=4, noise dim=32.

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

Table 17: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=8, noise dim=32.

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

Table 18: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=2.

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

Table 19: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=16.

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

Table 20: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=64.

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

Table 21: Performance of state-based DMC tasks for the compared methods in noise setting: noise std=1, noise dim=128.

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

Table 22: Performance of pixel-based DMC tasks for the compared methods with original clean background.

Task		SAC	DeepMDP	DBC	DBC-normed	MICo	RAP	SimSR
cartpole	balance	**994.5** ± 5.2	959.2 ± 36.7	957.6 ± 38.8	935.7 ± 71.0	981.9 ± 7.8	990.6 ± 8.0	985.1 ± 13.0
cartpole	balance_sparse	978.7 ± 46.1	928.9 ± 86.9	604.7 ± 618.1	**998.9** ± 2.5	990.8 ± 23.2	990.9 ± 21.5	994.7 ± 7.9
walker	stand	968.6 ± 9.0	961.1 ± 20.0	923.8 ± 63.4	963.2 ± 8.6	968.3 ± 7.7	**971.1** ± 6.3	875.2 ± 219.7
finger	spin	918.2 ± 173.4	**973.9** ± 29.9	837.5 ± 197.7	913.5 ± 179.2	845.5 ± 231.1	967.1 ± 40.7	918.6 ± 181.1
cartpole	swingup	855.3 ± 3.1	**859.7** ± 16.9	845.4 ± 40.6	827.8 ± 15.8	838.5 ± 50.1	859.2 ± 2.8	826.0 ± 56.1
ball_in_cup	catch	795.3 ± 468.4	793.2 ± 495.4	327.0 ± 433.9	899.5 ± 161.6	949.5 ± 13.7	967.6 ± 5.9	**972.6** ± 7.9
walker	walk	785.8 ± 129.8	955.9 ± 16.0	658.8 ± 184.6	941.9 ± 25.7	764.3 ± 18.9	936.6 ± 32.3	**957.2** ± 4.1
point_mass	easy	326.6 ± 556.4	483.3 ± 548.5	**769.8** ± 130.1	425.7 ± 503.9	500.3 ± 569.6	713.7 ± 494.4	180.9 ± 496.9
cartpole	swingup_sparse	767.8 ± 44.9	**790.2** ± 51.6	731.6 ± 43.5	749.4 ± 41.7	783.5 ± 43.9	730.9 ± 98.0	645.2 ± 448.2
reacher	easy	605.8 ± 198.5	135.9 ± 33.4	126.4 ± 119.2	193.0 ± 126.8	195.0 ± 79.3	**930.1** ± 46.2	85.2 ± 26.8
pendulum	swingup	410.8 ± 491.0	833.5 ± 7.9	14.0 ± 4.0	**839.2** ± 24.5	828.3 ± 19.9	829.7 ± 15.3	667.9 ± 462.7
cheetah	run	577.0 ± 162.4	398.4 ± 344.8	221.3 ± 265.4	296.4 ± 209.6	589.5 ± 102.0	**680.9** ± 214.1	674.3 ± 466.7
walker	run	304.3 ± 74.3	503.2 ± 171.3	142.9 ± 111.1	**578.2** ± 47.8	480.6 ± 32.8	530.1 ± 225.6	550.5 ± 74.8
hopper	hop	160.4 ± 21.1	**207.7** ± 46.0	17.9 ± 21.3	129.5 ± 90.3	177.0 ± 21.0	111.1 ± 50.9	112.9 ± 135.7

Table 23: Performance of pixel-based DMC tasks for the compared methods with grayscale images background.

Task		SAC	DeepMDP	DBC	DBC-normed	MICo	RAP	SimSR
cartpole	balance	985.8 ± 5.8	981.6 ± 20.0	729.1 ± 347.2	856.4 ± 299.6	853.6 ± 297.8	**989.1** ± 7.8	983.1 ± 10.3
cartpole	balance_sparse	998.7 ± 2.9	**999.7** ± 0.4	752.3 ± 520.5	761.8 ± 753.8	998.5 ± 1.8	995.0 ± 15.2	986.0 ± 21.1
walker	stand	955.7 ± 23.0	950.5 ± 38.0	822.1 ± 189.7	922.8 ± 84.9	952.6 ± 13.5	**961.4** ± 10.0	952.0 ± 20.2
finger	spin	972.2 ± 16.6	926.4 ± 155.9	558.5 ± 632.9	705.9 ± 499.9	906.8 ± 210.6	955.1 ± 66.0	**982.8** ± 4.8
cartpole	swingup	856.0 ± 13.3	842.1 ± 33.8	818.4 ± 39.6	764.8 ± 110.6	842.9 ± 25.9	**859.3** ± 13.1	492.2 ± 402.0
ball_in_cup	catch	761.5 ± 446.8	**789.7** ± 503.6	232.4 ± 293.1	630.3 ± 582.7	765.8 ± 435.5	786.6 ± 516.3	576.8 ± 583.3
walker	walk	640.0 ± 46.6	931.5 ± 66.0	502.5 ± 224.8	739.5 ± 495.3	712.2 ± 137.8	922.1 ± 13.1	**938.6** ± 26.4
point_mass	easy	179.2 ± 466.9	338.1 ± 573.1	393.9 ± 490.5	292.4 ± 441.2	533.4 ± 602.9	**538.1** ± 609.4	465.7 ± 414.0
cartpole	swingup_sparse	615.2 ± 653.1	625.8 ± 435.4	712.1 ± 48.5	590.6 ± 335.6	**794.8** ± 15.1	343.6 ± 487.4	789.0 ± 45.1
reacher	easy	588.6 ± 178.4	126.7 ± 83.6	116.4 ± 74.6	178.1 ± 90.9	208.0 ± 79.4	**966.7** ± 14.6	88.3 ± 38.1
pendulum	swingup	46.4 ± 107.3	335.6 ± 561.4	4.9 ± 7.0	342.1 ± 563.8	**540.2** ± 579.7	181.9 ± 442.2	345.3 ± 577.1
cheetah	run	376.5 ± 36.9	289.7 ± 200.4	210.6 ± 154.5	217.7 ± 244.2	364.8 ± 59.1	**409.7** ± 29.2	253.5 ± 285.7
walker	run	281.3 ± 52.4	267.1 ± 313.3	74.7 ± 209.4	310.3 ± 327.2	416.2 ± 33.1	**469.4** ± 144.8	404.5 ± 269.8
hopper	hop	**147.9** ± 19.9	100.4 ± 117.5	11.7 ± 31.2	71.7 ± 80.6	121.8 ± 89.5	82.2 ± 56.6	78.3 ± 133.5

Table 24: Performance of pixel-based DMC tasks for the compared methods with colored images background.

Task		SAC	DeepMDP	DBC	DBC-normed	MICo	RAP	SimSR
cartpole	balance	**984.1** ± 8.9	977.8 ± 13.1	978.4 ± 13.3	924.4 ± 54.5	978.6 ± 15.2	972.4 ± 21.8	978.7 ± 17.2
cartpole	balance_sparse	989.8 ± 15.1	**999.2** ± 1.1	787.1 ± 500.9	792.4 ± 502.7	996.1 ± 5.6	987.2 ± 29.5	994.7 ± 9.3
walker	stand	941.3 ± 53.9	949.7 ± 47.8	883.3 ± 166.4	958.7 ± 9.7	938.8 ± 57.4	**963.4** ± 8.1	948.4 ± 26.4
finger	spin	975.3 ± 17.4	902.4 ± 207.2	489.8 ± 583.7	670.4 ± 504.0	981.8 ± 2.9	786.8 ± 546.2	**984.4** ± 4.0
cartpole	swingup	**850.4** ± 9.2	844.0 ± 41.5	827.6 ± 20.5	805.0 ± 28.2	837.1 ± 23.9	845.8 ± 34.7	708.3 ± 371.2
ball_in_cup	catch	603.2 ± 598.5	933.1 ± 108.6	109.3 ± 18.7	586.0 ± 618.8	768.3 ± 437.4	**968.8** ± 5.4	445.6 ± 597.3
walker	walk	701.7 ± 158.9	570.0 ± 620.9	635.0 ± 37.4	828.2 ± 125.8	799.8 ± 162.9	907.9 ± 19.0	**938.2** ± 23.8
point_mass	easy	175.0 ± 479.6	524.0 ± 593.0	298.8 ± 510.6	469.1 ± 437.9	**709.5** ± 492.5	539.1 ± 600.5	643.1 ± 455.4
cartpole	swingup_sparse	611.4 ± 430.5	767.3 ± 124.5	717.6 ± 20.5	557.3 ± 390.6	783.0 ± 51.2	623.1 ± 407.5	**788.9** ± 47.4
reacher	easy	680.6 ± 146.4	170.6 ± 81.7	119.7 ± 35.3	185.8 ± 112.3	229.4 ± 78.3	**942.0** ± 20.6	101.6 ± 16.1
pendulum	swingup	102.6 ± 206.8	172.2 ± 459.8	1.9 ± 1.4	176.6 ± 469.1	**610.6** ± 638.5	390.4 ± 487.1	348.5 ± 584.7
cheetah	run	363.6 ± 31.9	373.9 ± 58.5	213.5 ± 226.9	285.1 ± 197.0	362.9 ± 36.6	**375.8** ± 59.6	205.9 ± 373.6
walker	run	269.1 ± 45.2	368.6 ± 268.3	140.9 ± 96.7	324.7 ± 273.9	385.2 ± 33.5	514.1 ± 68.3	**543.1** ± 43.0
hopper	hop	112.1 ± 127.8	**172.4** ± 43.9	10.3 ± 25.1	36.4 ± 114.2	63.2 ± 76.0	71.1 ± 83.2	118.0 ± 133.4

Table 25: Performance of pixel-based DMC tasks for the compared methods with grayscale video background.

Task		SAC	DeepMDP	DBC	DBC-normed	MICo	RAP	SimSR
cartpole	balance	976.9 ± 13.6	**987.8** ± 8.2	950.9 ± 24.7	880.2 ± 271.4	973.8 ± 16.2	979.1 ± 18.2	956.3 ± 59.8
cartpole	balance_sparse	992.8 ± 11.9	995.6 ± 10.7	866.2 ± 126.0	802.9 ± 547.4	**996.5** ± 2.9	974.5 ± 41.3	990.6 ± 13.2
walker	stand	961.5 ± 3.7	946.6 ± 64.9	643.8 ± 253.2	961.1 ± 3.1	947.8 ± 25.7	**971.0** ± 3.9	957.0 ± 8.7
finger	spin	976.9 ± 11.8	**982.7** ± 2.9	0.0 ± 0.0	974.8 ± 13.1	979.9 ± 7.7	971.4 ± 21.4	902.6 ± 261.3
cartpole	swingup	854.9 ± 24.2	866.5 ± 10.9	826.8 ± 43.7	410.4 ± 410.3	854.2 ± 5.8	863.5 ± 7.8	**868.0** ± 22.2
ball_in_cup	catch	961.6 ± 3.3	690.1 ± 124.2	105.1 ± 30.0	862.4 ± 172.6	935.1 ± 22.4	**964.1** ± 10.6	941.1 ± 52.9
walker	walk	650.4 ± 191.3	869.2 ± 25.5	146.5 ± 84.5	885.0 ± 46.9	763.3 ± 258.4	**913.6** ± 33.1	907.8 ± 48.7
point_mass	easy	**884.4** ± 23.0	712.3 ± 489.7	166.9 ± 452.1	26.5 ± 71.8	870.0 ± 73.0	449.6 ± 372.0	545.3 ± 596.9
cartpole	swingup_sparse	650.8 ± 452.7	646.3 ± 450.0	0.0 ± 0.0	305.9 ± 378.9	**802.3** ± 34.2	369.5 ± 401.1	489.9 ± 555.5
reacher	easy	313.8 ± 56.9	187.3 ± 57.3	154.3 ± 115.1	167.0 ± 31.8	206.8 ± 84.0	**972.0** ± 5.1	75.5 ± 10.5
pendulum	swingup	202.5 ± 450.9	**839.3** ± 12.3	31.9 ± 85.1	63.1 ± 232.0	707.7 ± 347.0	705.9 ± 325.7	833.7 ± 17.2
cheetah	run	359.7 ± 27.3	341.4 ± 67.1	75.8 ± 67.0	174.0 ± 197.1	371.9 ± 12.8	**405.0** ± 22.2	342.4 ± 21.7
walker	run	244.8 ± 45.7	175.8 ± 106.2	63.1 ± 17.1	180.3 ± 109.5	393.4 ± 16.2	**544.0** ± 26.3	270.5 ± 190.6
hopper	hop	**121.5** ± 84.7	17.3 ± 45.9	0.1 ± 0.1	38.6 ± 67.2	117.7 ± 82.5	77.3 ± 53.8	1.4 ± 1.1

Table 26: Performance of pixel-based DMC tasks for the compared methods with colored video background.

Task		SAC	DeepMDP	DBC	DBC-normed	MICo	RAP	SimSR
cartpole	balance	980.3 ± 8.7	**983.2** ± 21.8	904.4 ± 76.5	774.3 ± 355.0	952.9 ± 43.0	949.0 ± 47.2	982.0 ± 9.8
cartpole	balance_sparse	985.1 ± 13.8	995.0 ± 8.5	759.3 ± 197.7	972.2 ± 60.5	978.4 ± 63.0	973.3 ± 66.0	**997.6** ± 6.7
walker	stand	**961.8** ± 10.8	916.2 ± 128.5	534.5 ± 347.5	951.0 ± 5.3	943.3 ± 17.7	960.0 ± 12.1	957.7 ± 5.6
finger	spin	962.1 ± 27.0	**982.2** ± 4.6	163.9 ± 521.6	905.2 ± 177.5	978.8 ± 12.7	860.1 ± 331.9	908.8 ± 212.0
cartpole	swingup	848.4 ± 13.9	**861.4** ± 169.4	753.1 ± 41.3	293.9 ± 337.8	849.4 ± 10.1	857.1 ± 30.5	830.5 ± 56.6
ball_in_cup	catch	**937.3** ± 57.6	389.0 ± 264.6	111.1 ± 67.4	762.4 ± 411.0	877.2 ± 113.0	758.7 ± 639.9	891.7 ± 78.2
walker	walk	684.6 ± 151.0	667.6 ± 452.6	100.6 ± 144.0	651.9 ± 458.0	845.3 ± 83.9	**921.6** ± 30.3	912.6 ± 26.2
point_mass	easy	888.2 ± 13.2	366.8 ± 588.0	1.6 ± 2.0	281.8 ± 656.0	**889.7** ± 15.3	29.2 ± 41.3	184.4 ± 476.5
cartpole	swingup_sparse	705.6 ± 155.5	579.5 ± 429.8	0.0 ± 0.0	45.7 ± 125.3	**733.4** ± 128.6	12.4 ± 18.1	673.6 ± 228.7
reacher	easy	261.5 ± 118.5	165.6 ± 89.1	180.0 ± 75.4	208.3 ± 96.8	149.8 ± 31.1	**960.2** ± 26.8	95.2 ± 46.0
pendulum	swingup	80.9 ± 67.7	486.7 ± 434.3	6.2 ± 8.1	338.5 ± 551.4	**710.4** ± 429.8	439.2 ± 460.9	656.2 ± 451.1
cheetah	run	348.1 ± 6.4	304.0 ± 40.3	53.6 ± 60.9	223.0 ± 49.8	344.6 ± 30.2	**396.1** ± 19.5	319.9 ± 21.8
walker	run	244.1 ± 70.8	115.8 ± 103.8	54.9 ± 23.3	148.5 ± 153.7	367.0 ± 33.4	**492.7** ± 135.4	168.8 ± 170.5
hopper	hop	**136.3** ± 19.5	23.0 ± 36.9	0.5 ± 1.0	27.8 ± 46.6	106.2 ± 80.3	40.4 ± 43.1	27.5 ± 50.1

Table 27: Performance of state-based DMC tasks for the compared methods with IID Gaussian noise background.

Task		SAC	DeepMDP	DBC	DBC-normed	MICo	RAP	SimSR
cartpole	balance	975.0 ± 12.3	973.0 ± 40.5	951.0 ± 18.1	982.6 ± 26.8	959.6 ± 23.9	981.3 ± 8.0	**988.4** ± 6.1
cartpole	balance_sparse	984.8 ± 29.0	999.8 ± 0.4	855.4 ± 118.4	**1000.0** ± 0.0	873.7 ± 171.8	962.5 ± 61.3	911.9 ± 105.7
walker	stand	951.5 ± 14.2	951.2 ± 37.9	896.1 ± 311.1	373.2 ± 397.6	941.0 ± 33.3	**959.0** ± 9.2	936.9 ± 18.3
finger	spin	720.5 ± 250.6	587.1 ± 663.7	572.1 ± 648.5	**945.1** ± 67.5	814.5 ± 411.0	863.8 ± 306.2	786.3 ± 545.8
cartpole	swingup	655.5 ± 282.0	835.9 ± 43.3	796.4 ± 64.1	737.3 ± 258.3	804.7 ± 34.6	**851.6** ± 10.4	811.7 ± 88.7
ball_in_cup	catch	562.3 ± 351.7	**966.1** ± 5.6	202.3 ± 245.4	737.8 ± 636.2	804.7 ± 187.8	898.8 ± 45.0	786.1 ± 494.1
walker	walk	608.8 ± 120.0	**936.0** ± 34.6	411.0 ± 447.9	685.2 ± 695.2	708.4 ± 199.5	907.4 ± 43.2	933.2 ± 37.8
point_mass	easy	5.5 ± 13.5	3.4 ± 5.3	3.2 ± 6.7	3.0 ± 4.1	0.7 ± 5.7	**321.7** ± 546.1	171.9 ± 475.9
cartpole	swingup_sparse	0.0 ± 0.0	249.6 ± 1073.8	261.2 ± 444.3	257.9 ± 439.8	121.9 ± 320.2	0.7 ± 0.6	**469.9** ± 534.3
reacher	easy	275.3 ± 121.8	168.6 ± 52.5	178.7 ± 57.1	237.7 ± 116.6	186.3 ± 54.5	**961.6** ± 15.6	83.2 ± 9.8
pendulum	swingup	**657.6** ± 455.6	207.6 ± 657.0	9.1 ± 10.3	215.2 ± 666.8	501.5 ± 540.4	232.7 ± 590.1	175.8 ± 467.3
cheetah	run	306.5 ± 33.8	91.8 ± 171.3	214.6 ± 148.5	74.2 ± 199.2	295.9 ± 16.8	**309.7** ± 35.5	303.1 ± 120.3
walker	run	205.6 ± 25.5	411.2 ± 273.8	26.7 ± 1.8	242.6 ± 239.6	304.9 ± 80.4	330.6 ± 227.2	**487.5** ± 36.3
hopper	hop	47.3 ± 82.6	28.4 ± 78.6	0.1 ± 0.1	22.0 ± 60.5	45.6 ± 77.3	**68.2** ± 47.9	52.1 ± 96.5

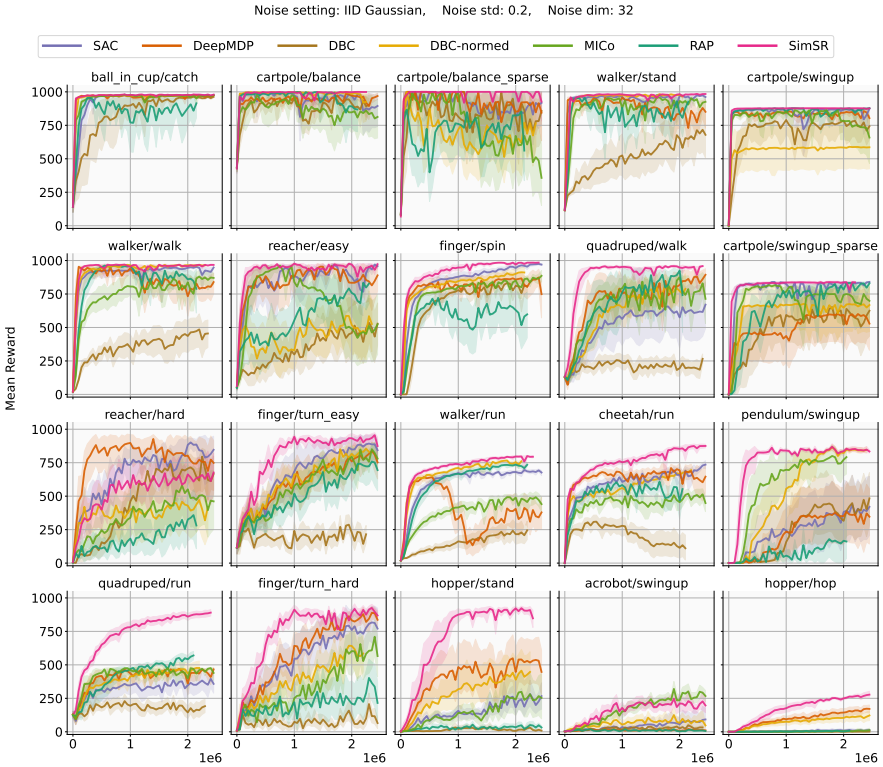

Figure 13: Performance on individual state-based tasks.

Figure 14: Performance on individual state-based tasks.

Figure 15: Performance on individual state-based tasks.

Figure 16: Performance on individual state-based tasks.

Figure 17: Performance on individual state-based tasks.

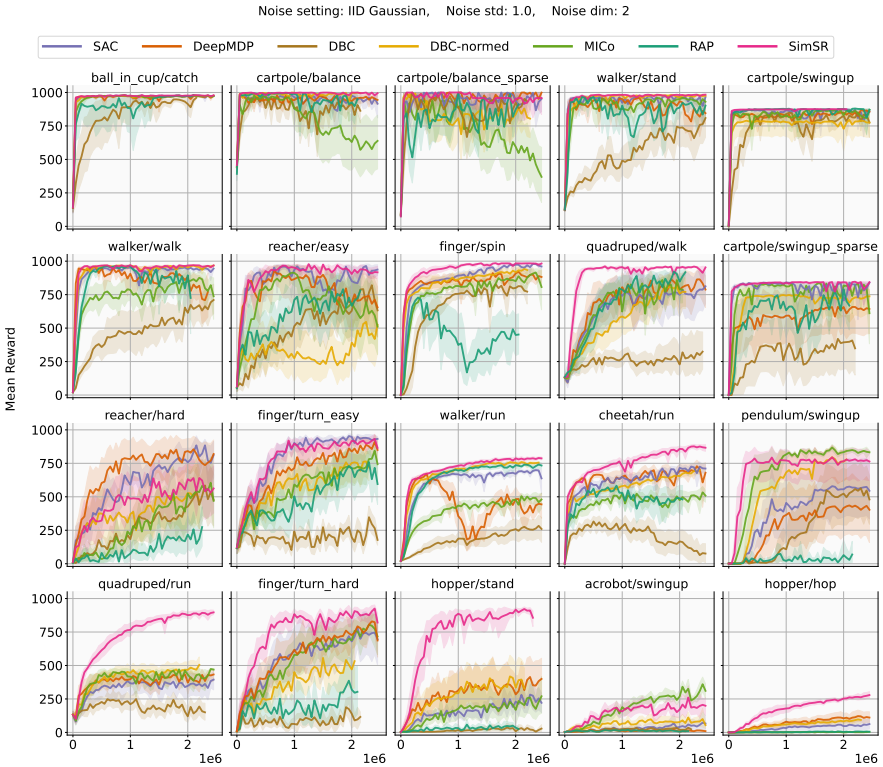

Figure 18: Performance on individual state-based tasks.

Figure 19: Performance on individual state-based tasks.

Figure 20: Performance on individual state-based tasks.

Figure 21: Performance on individual state-based tasks.

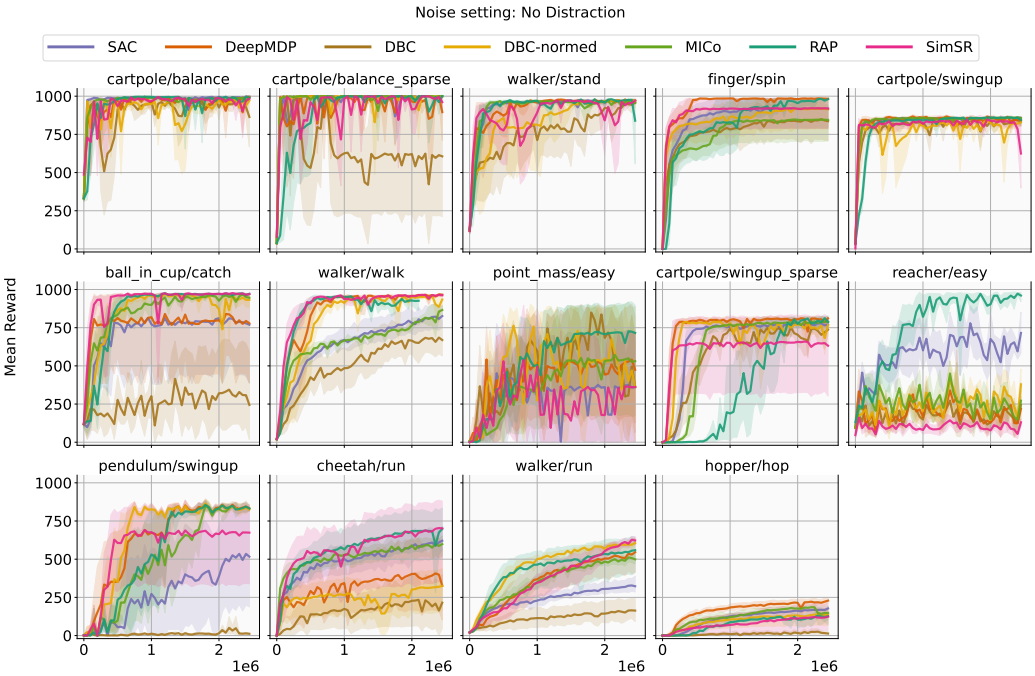

Figure 22: Performance on individual pixel-based tasks.

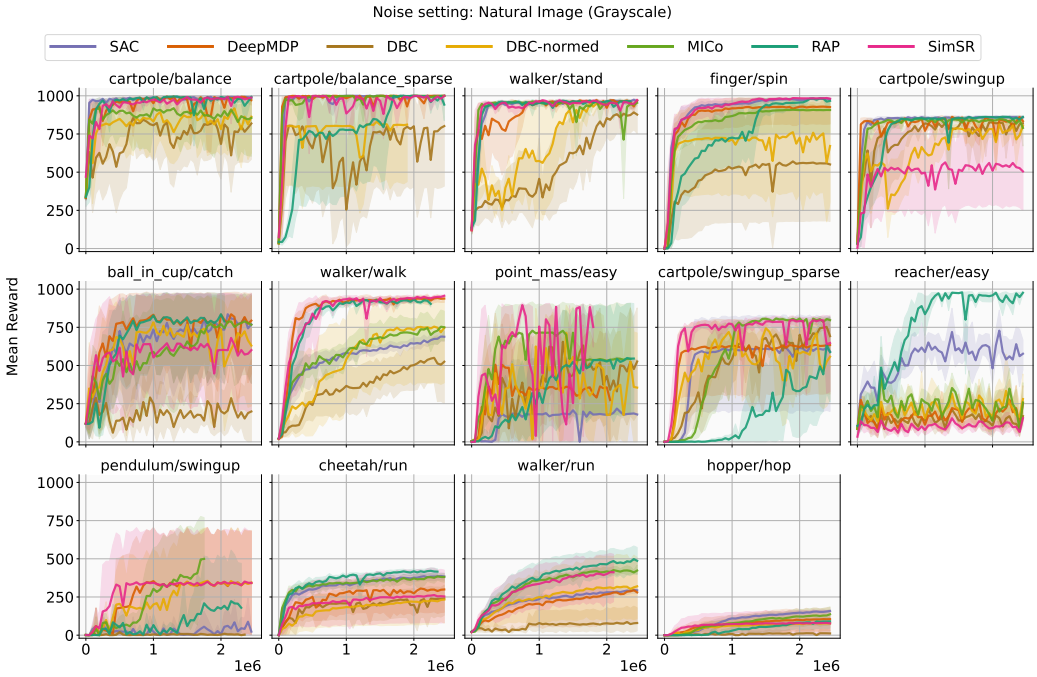

Figure 23: Performance on individual pixel-based tasks.

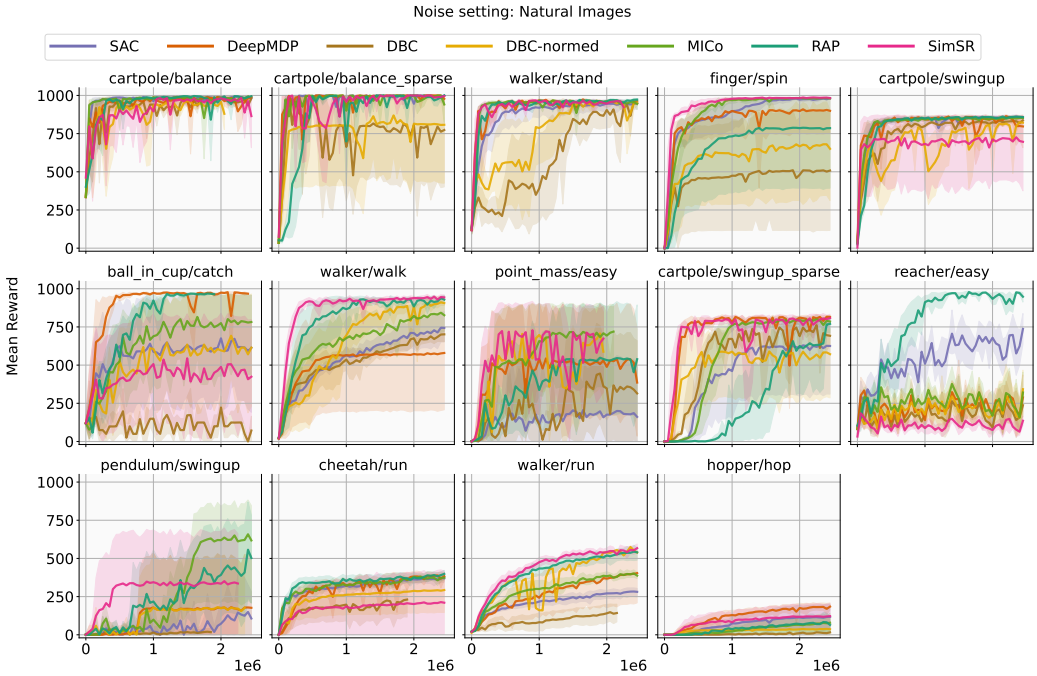

Figure 24: Performance on individual pixel-based tasks.

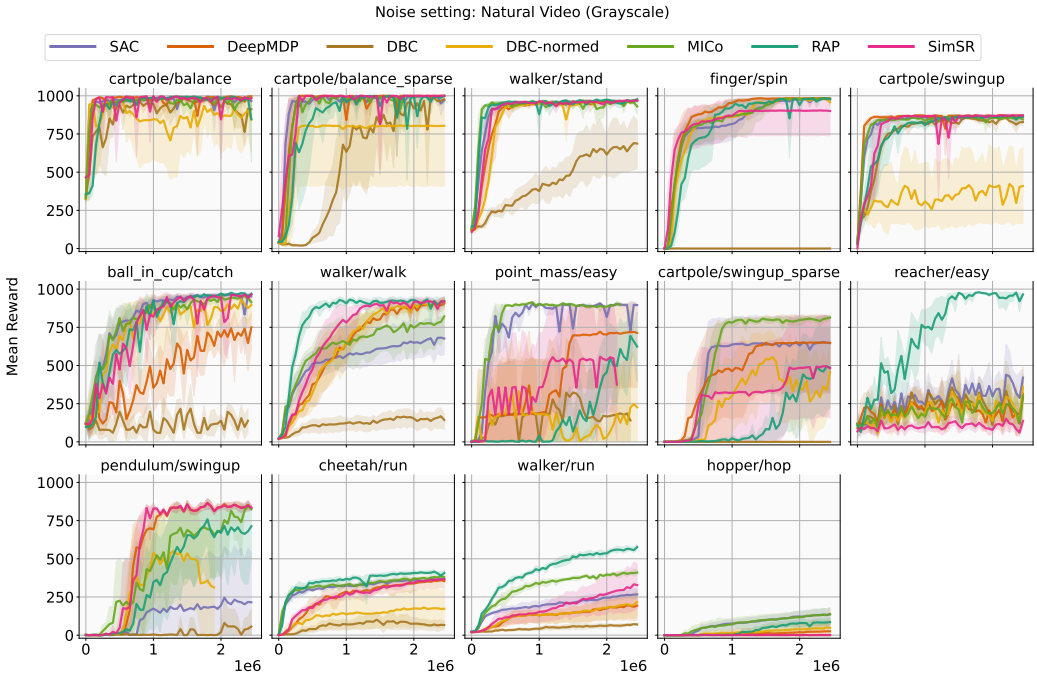

Figure 25: Performance on individual pixel-based tasks.

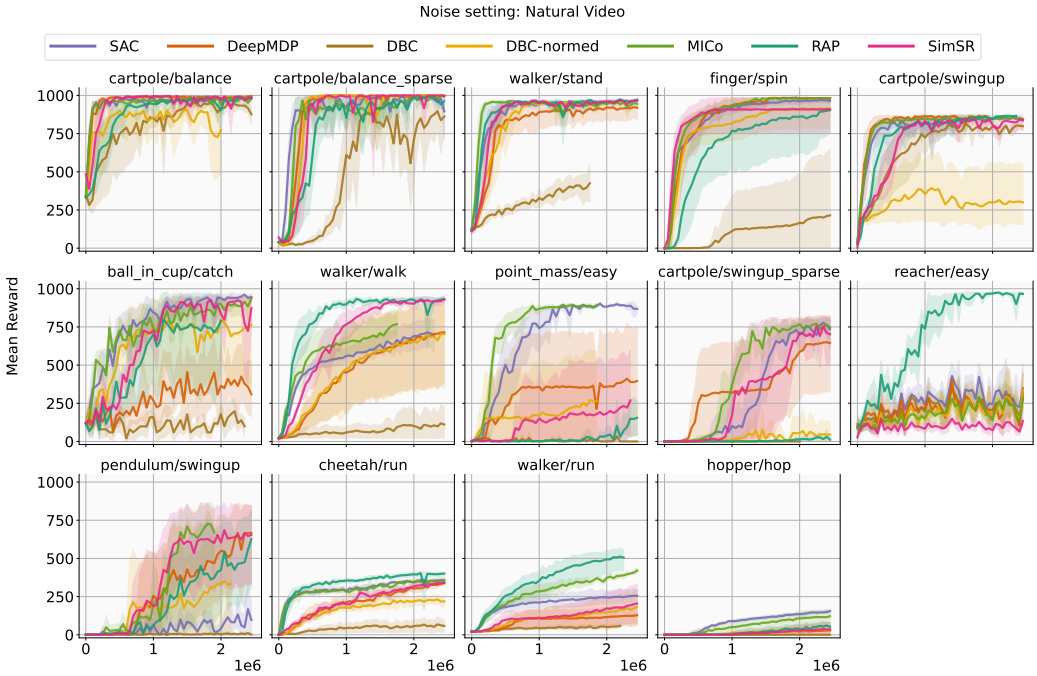

Figure 26: Performance on individual pixel-based tasks.

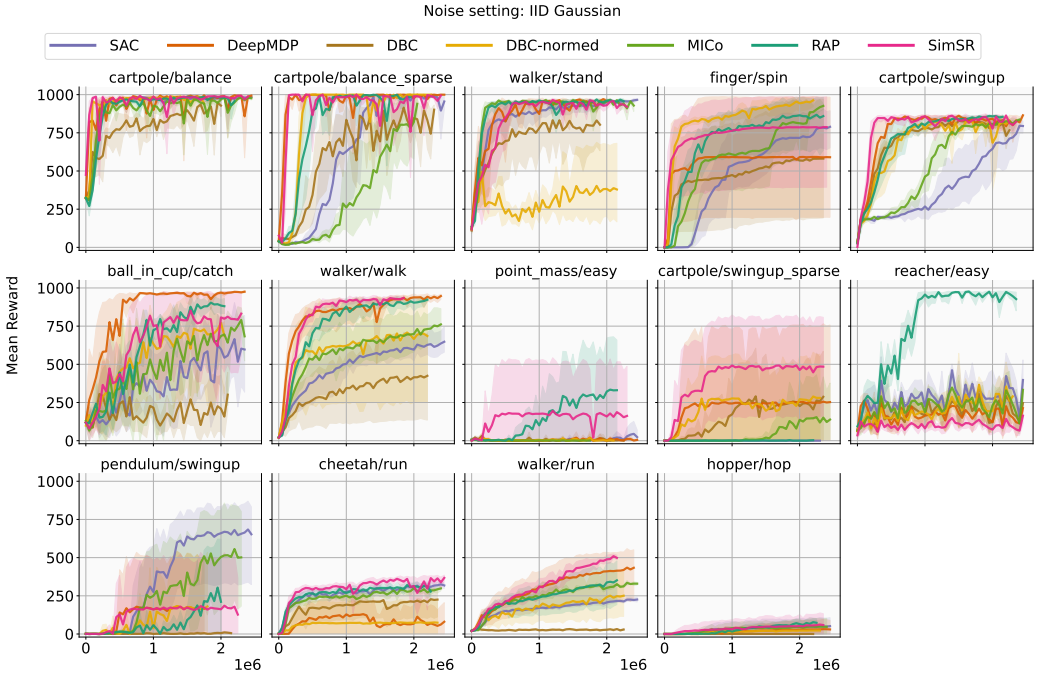

Figure 27: Performance on individual pixel-based tasks.

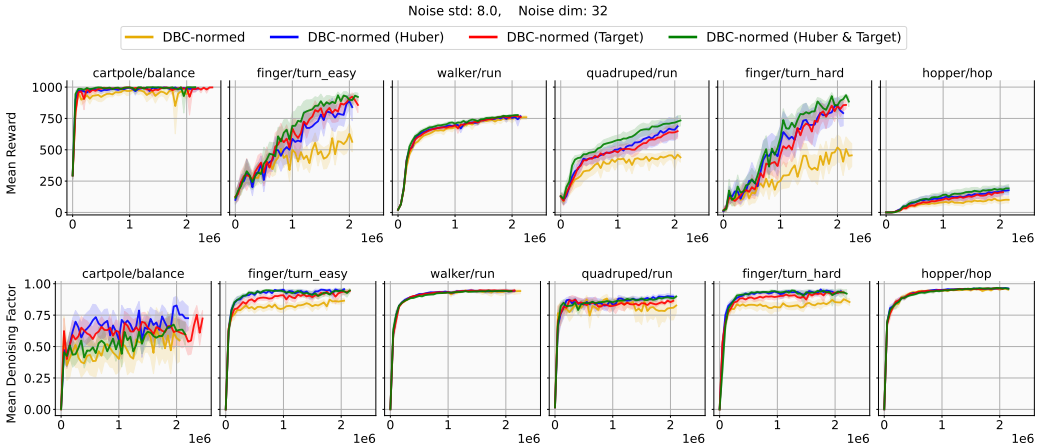

Figure 28: Case study on 6 state-based DMC tasks on **DBC-normed with LayerNorm** and its variants: **applying the target trick** (DBC-normed (Target)), **using Huber loss** (DBC-normed (Huber)) instead of MSE as the metric loss, and **applying both** (DBC-normed (Huber & Target)). X-axis is the environmental step.

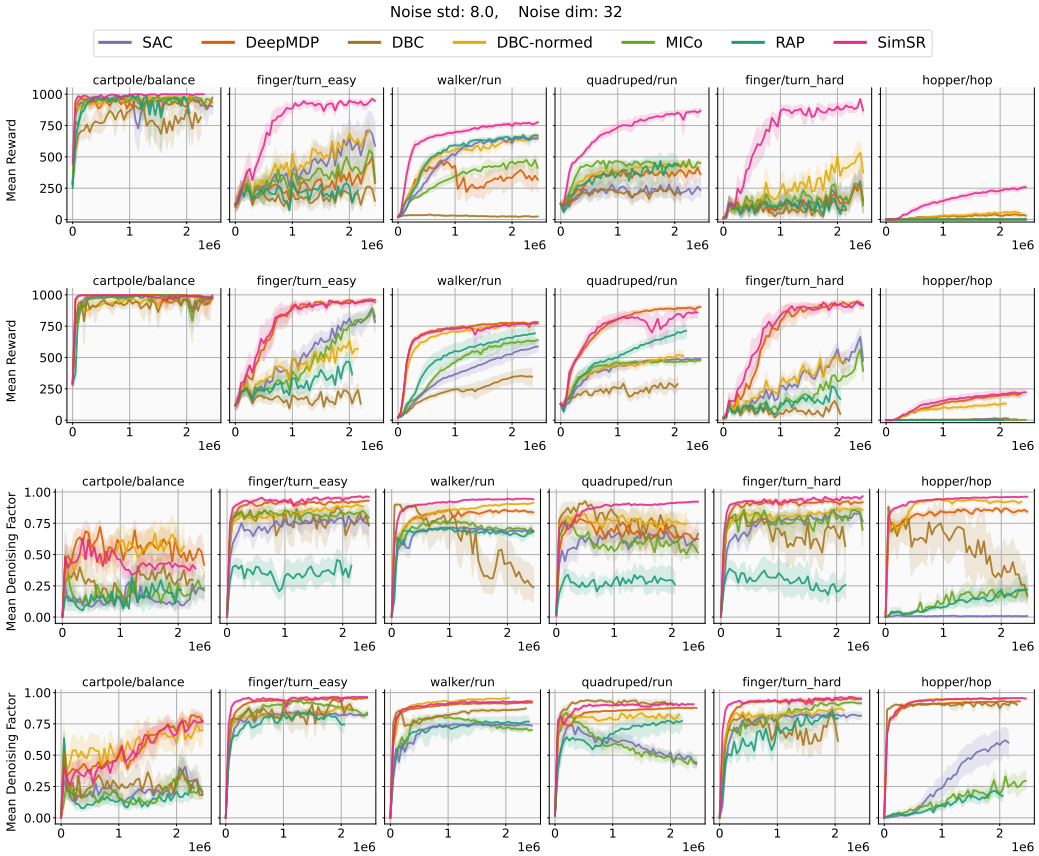

Figure 29: Case study on six DMC state-based tasks examining the effects of **including LayerNorm** (the first and third vs. second and fourth rows). X-axis stands for the environmental step. See Table 2 for a tabular presentation.

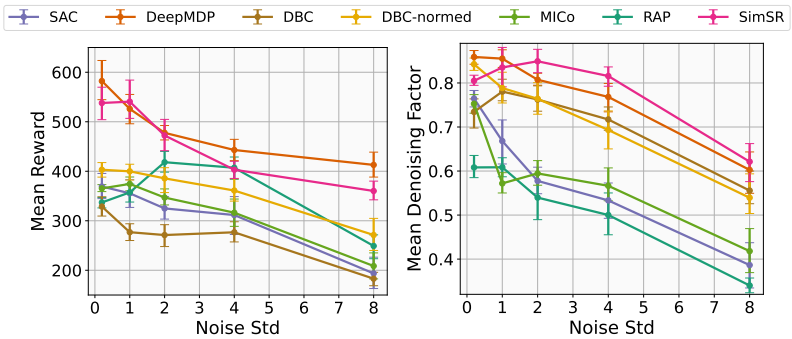

Figure 30: **Aggregated reward (left) and denoising factor (right)** of methods on **IID Gaussian noise with random projection** setting, varying noise standard deviation, in the 6 selected state-based tasks.

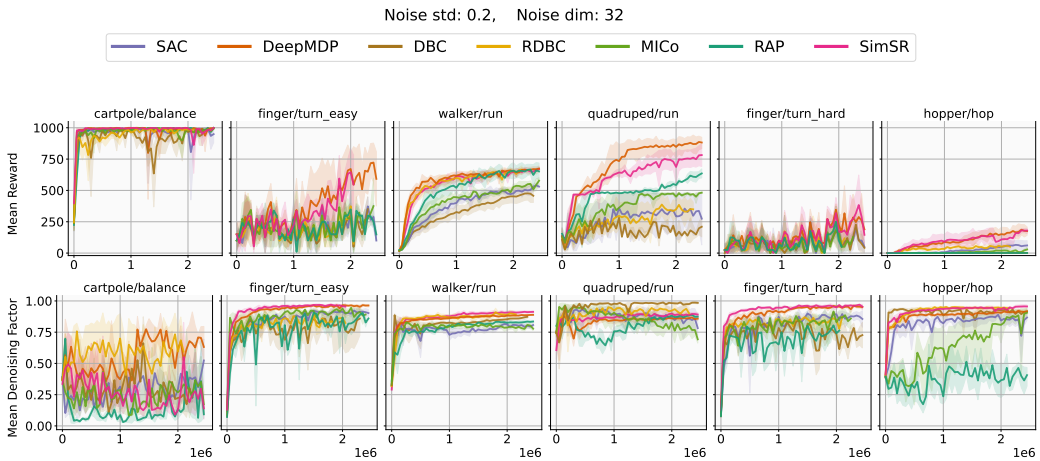

Figure 31: Performance on individual state-based tasks under the IID Gaussian noise with random projection setting.

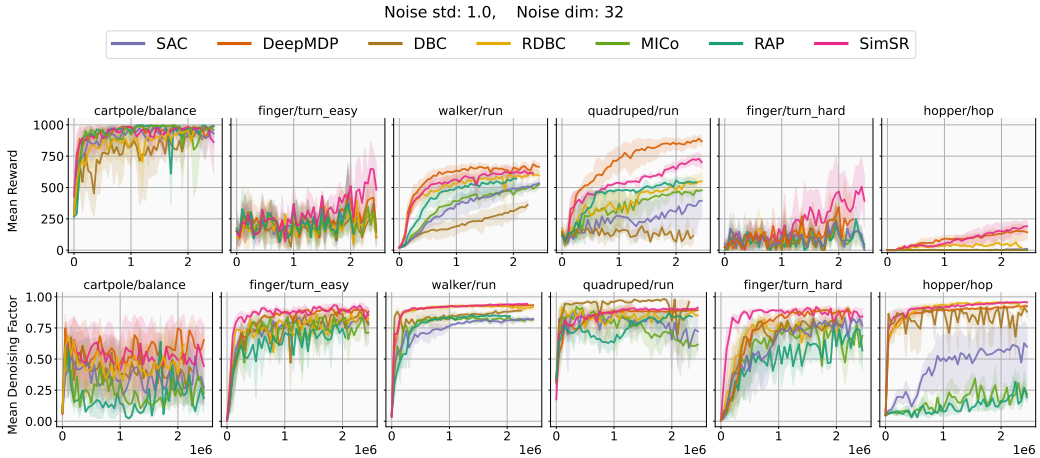

Figure 32: Performance on individual state-based tasks under the IID Gaussian noise with random projection setting.

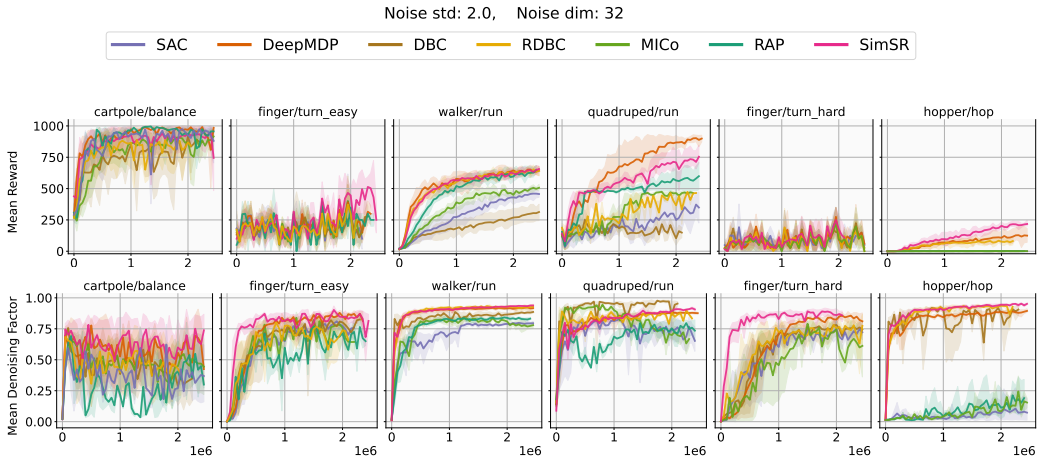

Figure 33: Performance on individual state-based tasks under the IID Gaussian noise with random projection setting.

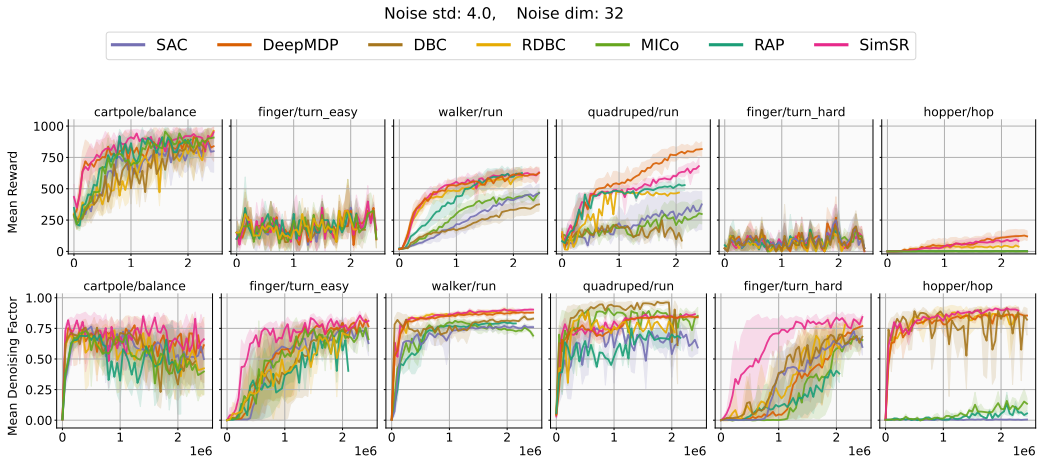

Figure 34: Performance on individual state-based tasks under the IID Gaussian noise with random projection setting.

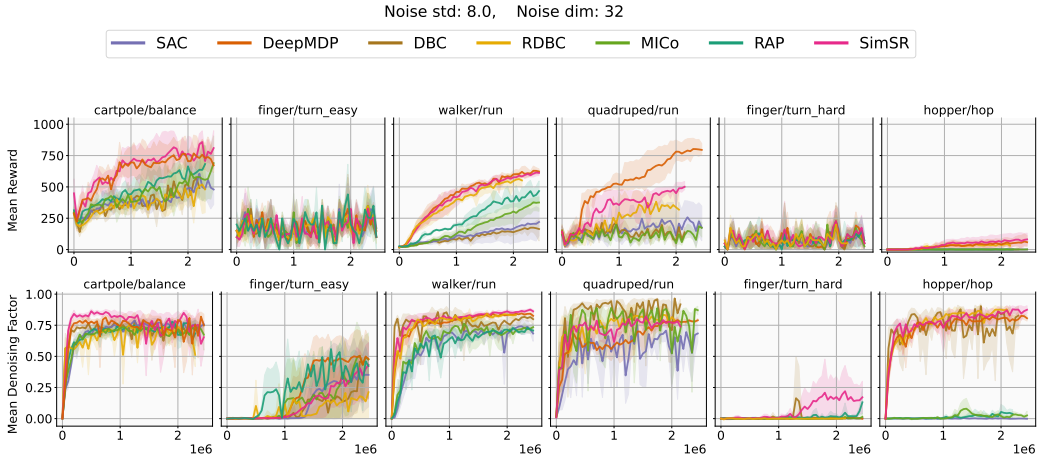

Figure 35: Performance on individual state-based tasks under the IID Gaussian noise with random projection setting.

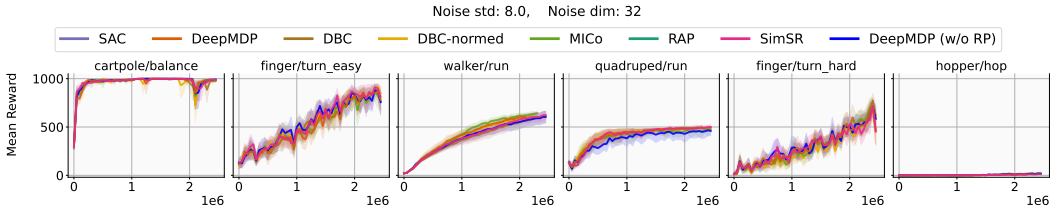

Figure 36: Reward curves for the isolated metric estimation setting, using the SAC agent with LayerNorm. Since this setting isolates metric learning, all methods differ only in how the metric encoder $\tilde{\phi}$ is shaped, and thus similar performance across methods is expected. This also ensures that the collected data distribution remains consistent when evaluating different metric learning methods.

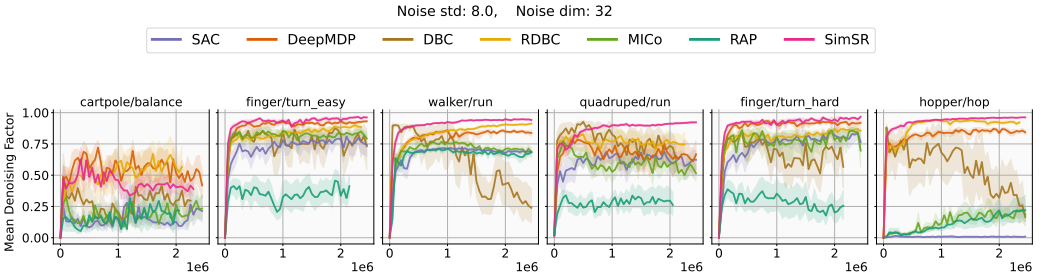

Figure 37: DF for the agent encoder ϕ (co-trained with RL in Sec. 5.1) without LayerNorm.

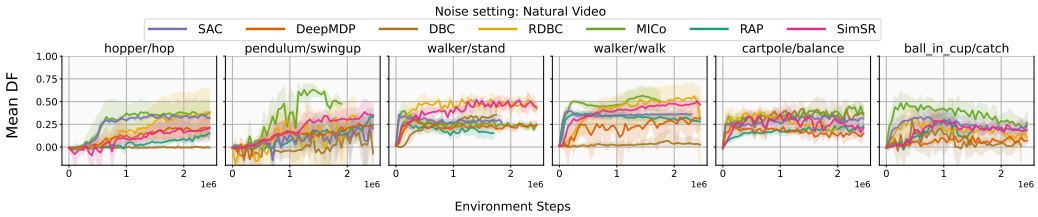

Figure 38: Denoising factor curves on six pixel-based DMC tasks under the **OOD evaluation setting**, measured on the agent encoder ϕ.

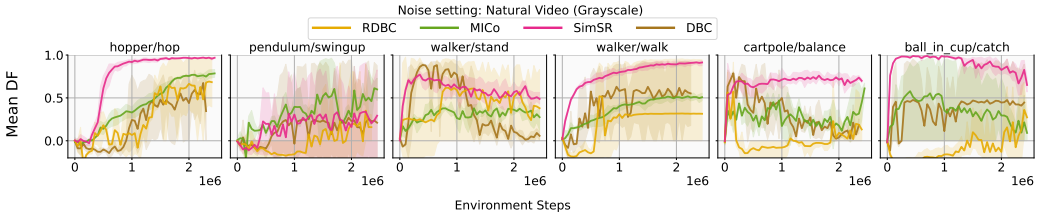

Figure 39: Denoising factor curves on six pixel-based DMC tasks under the **ID evaluation setting**, measured on the isolated metric encoder $\tilde{\phi}$.

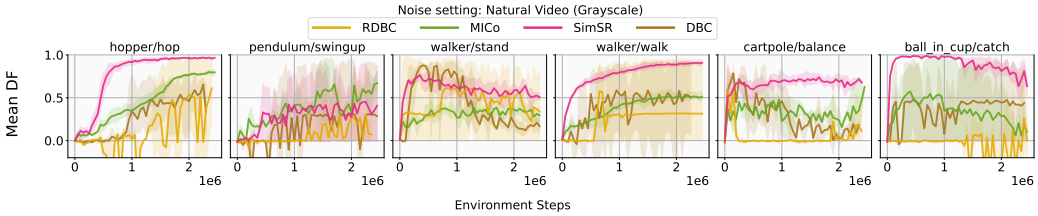

Figure 40: Denoising factor curves on six pixel-based DMC tasks under the **OOD evaluation setting**, measured on the isolated metric encoder $\tilde{\phi}$.

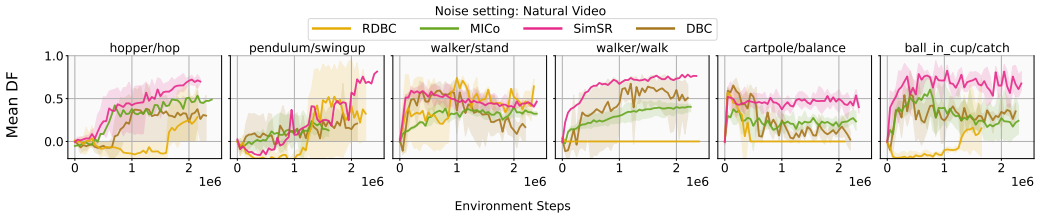

Figure 41: Denoising factor curves on six pixel-based DMC tasks under the **ID evaluation setting**, measured on the isolated metric encoder $\tilde{\phi}$.

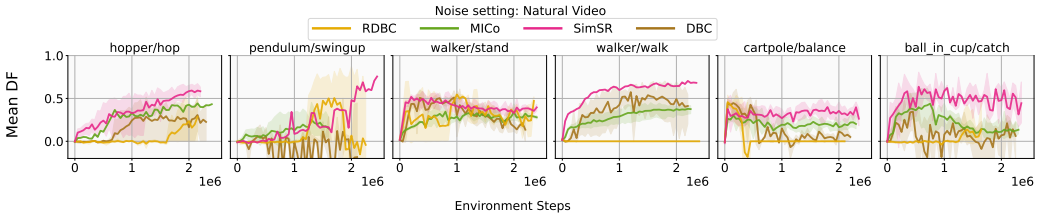

Figure 42: Denoising factor curves on six pixel-based DMC tasks under the **OOD evaluation setting**, measured on the isolated metric encoder $\tilde{\phi}$.

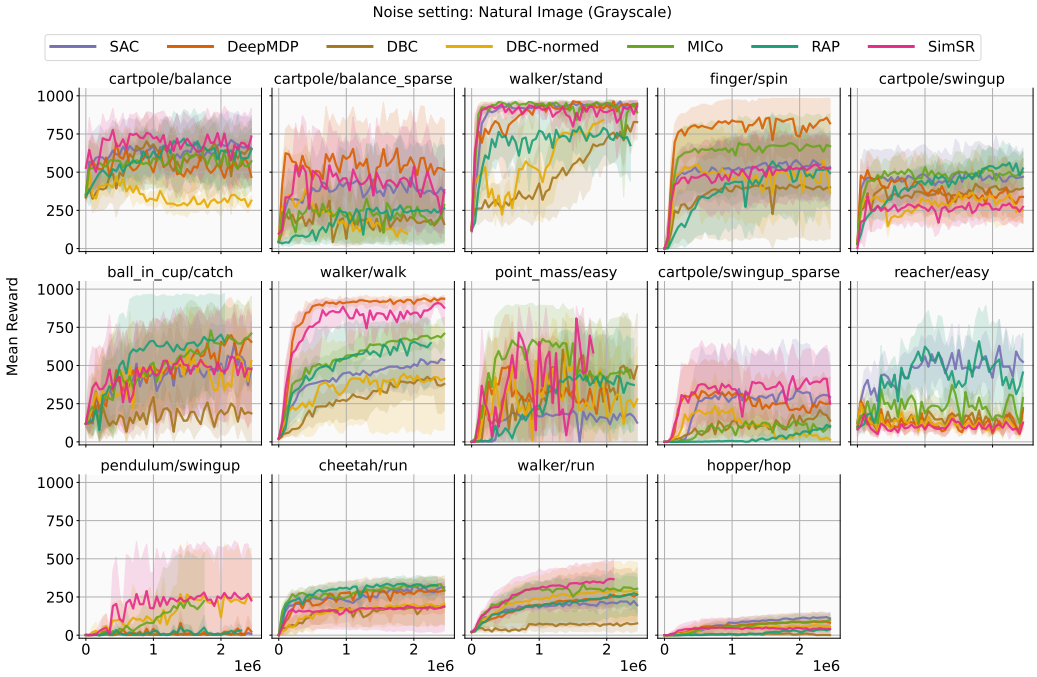

Figure 43: Per-task OOD generalization reward curves for pixel-based DMC tasks.

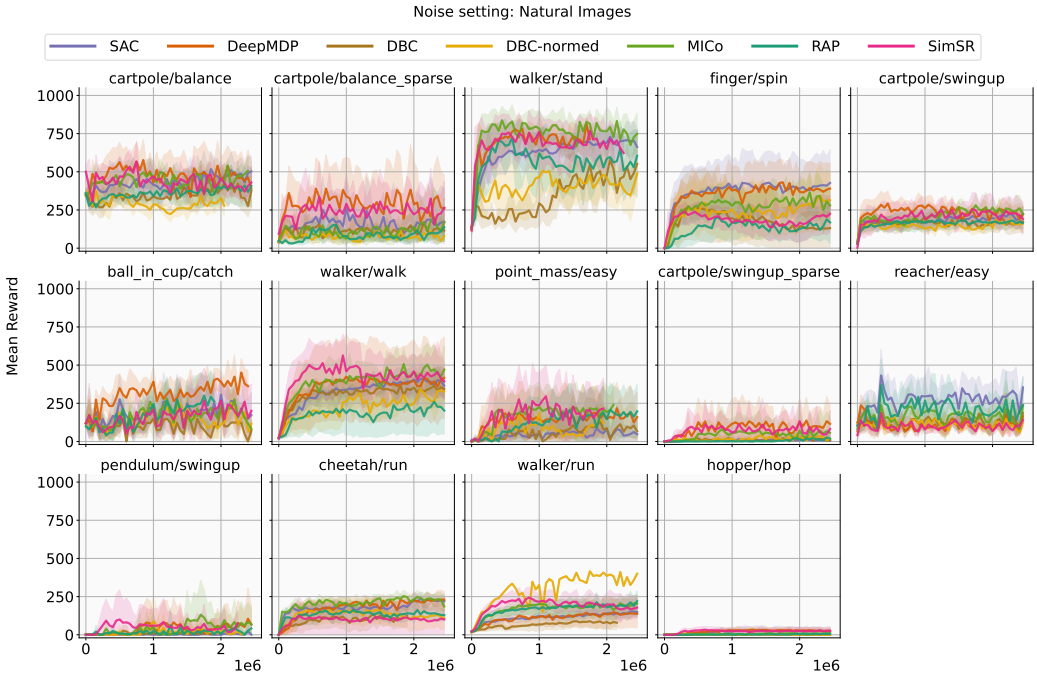

Figure 44: Per-task OOD generalization reward curves for pixel-based DMC tasks.

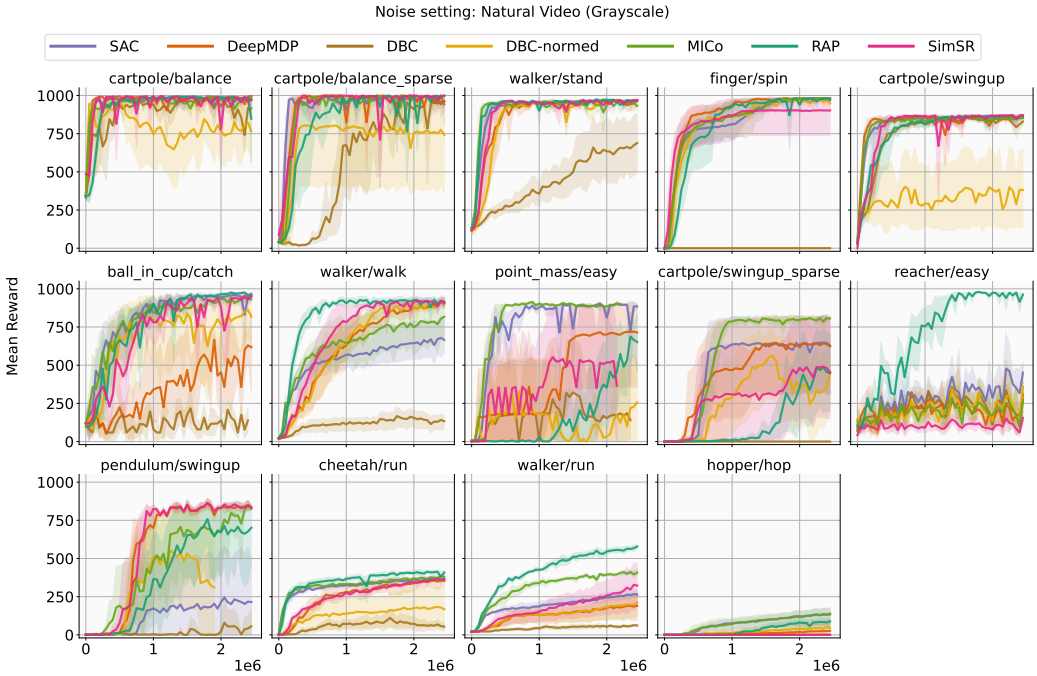

Figure 45: Per-task OOD generalization reward curves for pixel-based DMC tasks.

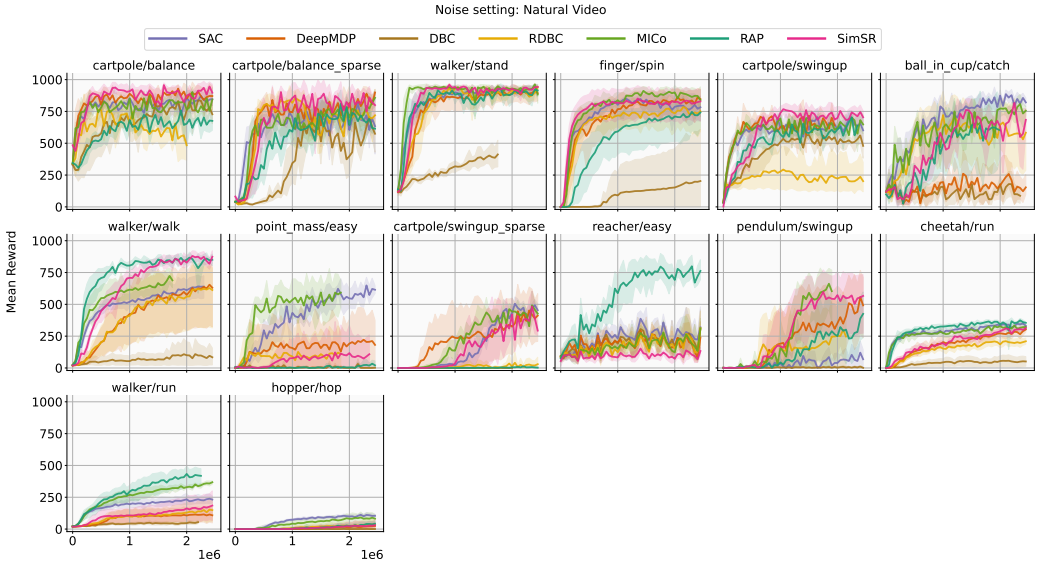

Figure 46: Per-task OOD generalization reward curves for pixel-based DMC tasks.

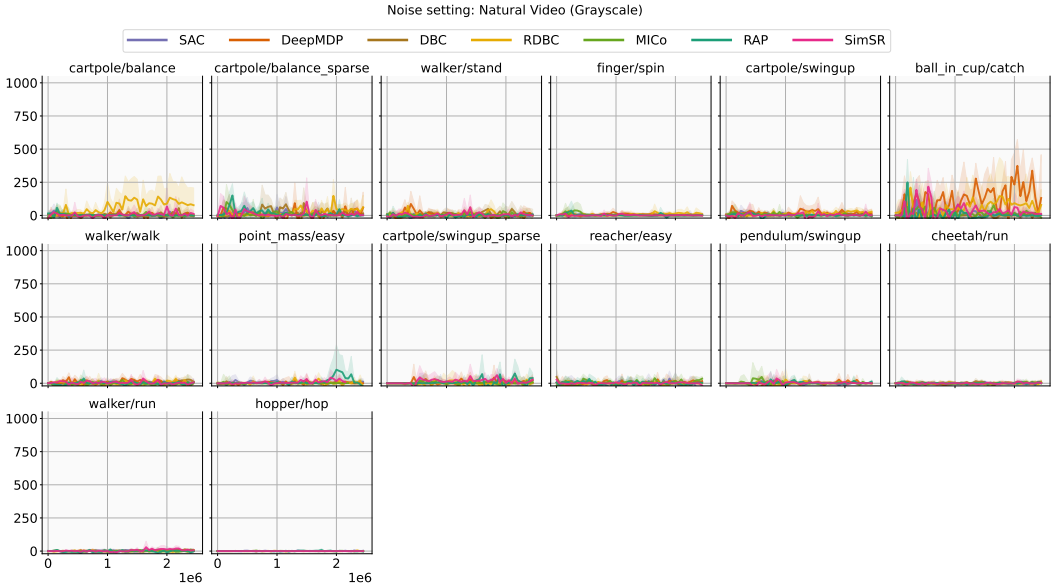

Figure 47: Pixel-based DMC per-task generalization **reward difference** curves.

Figure 48: Pixel-based DMC per-task generalization **reward difference** curves.