Mila

Codebase

Understanding Effectiveness of Learning Behavioral *Metrics* in Deep Reinforcement *Learning*

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





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

About me

Ziyan "Ray" Luo

Research Interests:

- Abstraction in RL, HRL
- Metric learning in RL
- Formal Verification (CHC)

Hobbies:

- Composing music (for video games)
- Ball sports (ping pong, tennis, billiard)
- Animals

https://zyluo.netlify.app/

https://soundcloud.com/sunsetray

Background

Metric learning provides a first-principle method for state abstraction.

Proprioceptive states

Motivation: State Abstraction in RL

Scaling RL to **high-dimensional, distraction-rich** domains remains challenging

=>

=>

[1.15,
2.53,
3.45,
-2.02,
...]

Observations

a compact state

Distracting DMC: Over **90%** pixels are **task-irrelevant**

Percentage of Distracting (Task-irrelevant) Pixels

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%

DMC with distraction

Noise can be structured!
E.g., temporally dependent

State Abstraction

State may be high-dimensional, e.g., pixel input, torque control parameters

- A good abstraction gives a good problem formulation (George Konidaris, 2019)
- **Benefits**: sample efficiency, generalization/robustness, computation efficiency, better value estimation, …
- **State abstraction**: traditionally by **partitioning** the states space using **equivalence relation**
- How to define states as **equivalent?**

A “lossless compression” of an MDP

Examples of State Abstraction

Traditionally, it is done by *aggregating* the states with *exact standards*

- For example, **two states** are deemed equivalent if:

$$\forall a \in \mathcal{A}, \quad \mathcal{P}(\cdot \mid x_1, a) = \mathcal{P}(\cdot \mid x_2, a),$$
$$\mathcal{R}(x_1, a) = \mathcal{R}(x_2, a)$$

Bisimulation

$$\forall a \in \mathcal{A}, \forall \pi, \quad Q^\pi(x_1, a) = Q^\pi(x_2, a)$$
$$\forall a \in \mathcal{A}, \quad Q^*(x_1, a) = Q^*(x_2, a)$$

ϕ_{Q^*} ϕ_{Q^π}

Value-preserving Abstraction

$$\mathcal{R}^\pi(x_1) = \mathcal{R}^\pi(x_2), \quad \mathcal{P}^\pi(\cdot \mid x_1) = \mathcal{P}^\pi(\cdot \mid x_2)$$

Policy-dependent Bisimulation

$$\mathcal{R}^\pi(x) := \mathbb{E}_{a\sim\pi(\cdot|x)}[\mathcal{R}(x, a)]$$
$$\mathcal{P}^\pi(\cdot \mid x) := \mathbb{E}_{a\sim\pi(\cdot|x)}[\mathcal{P}(\cdot \mid x, a)]$$

$$\pi^*(\cdot \mid x_1) = \pi^*(\cdot \mid x_2),$$

Denoised MDPs (Wang et al. 2023)

Towards a Unified Theory of State Abstraction for MDPs, Li et al., 2006

Bisimulation Relation

Given an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, let

$$\mathcal{S}/\sim = \{ C \subseteq \mathcal{S} \mid C \neq \emptyset, \forall s_1, s_2 \in C : s_1 \sim s_2 \}$$

denote the set of equivalence classes under $\sim$ (each C is one class). A bisimulation relation $\sim \subseteq \mathcal{S} \times \mathcal{S}$ is the **largest relation** such that for any $s_1, s_2 \in \mathcal{S}$ with $s_1 \sim s_2$ and $\forall a \in \mathcal{A}$:

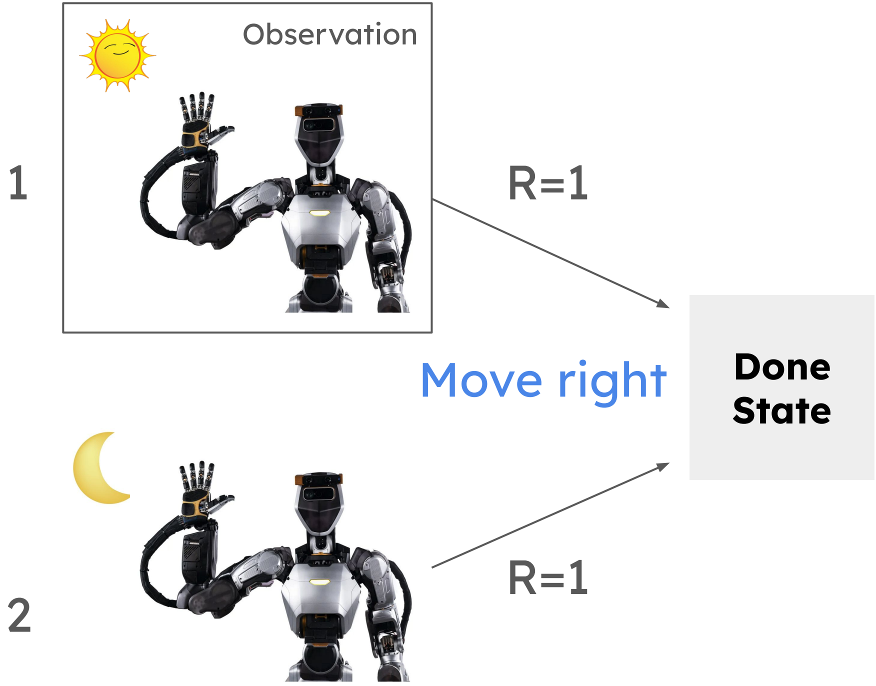
$$R(s_1, a) = R(s_2, a),$$

$$\forall C \in \mathcal{S}/\sim: \sum_{s' \in C} P(s' \mid s_1, a) = \sum_{s' \in C} P(s' \mid s_2, a).$$

Thus, bisimilar states yield identical immediate rewards and transition probabilities over each equivalence class C.

Illustrative example of bisimulation relation

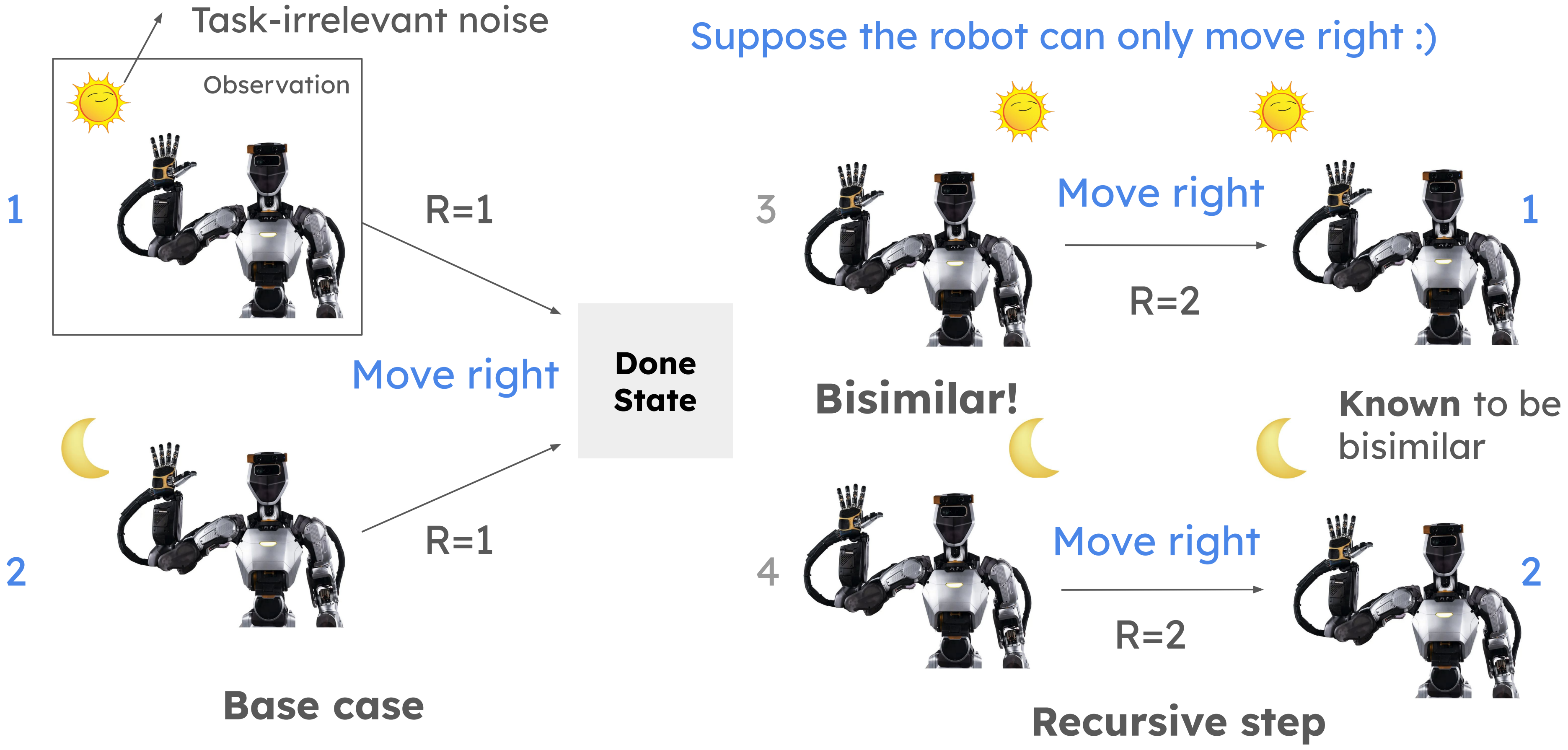
Suppose the robot can only move right :)



Bisimilar!

Base case

Illustrative example of bisimulation relation



Limitations of State Aggregation

State may be high-dimensional, e.g., pixel input, torque control parameters

- State abstraction: traditionally by **partitioning** the states space using **equivalence relation**
- Bisimulation relation: **reward & transition equivalence (under same actions)**
- **Dichotomy**: two states are either bisimilar or not
- Doesn’t work for high-dimensional observations
- Hard to compute online

A “lossless compression” of an MDP

Metrics: *Relaxing* State Aggregation

Bisimulation metrics (BSMs) **relax** the bisimulation relation by allowing smooth variation based on differences in reward and transition dynamics. It quantifies behavioral similarity between observations:

$$d^{\sim}(x_1, x_2) = \max_{a \in \mathcal{A}} \Big(c_R \, |\mathcal{R}(x_1, a) - \mathcal{R}(x_2, a)| + c_T \, \mathcal{W}_1(d^{\sim})\big(\mathcal{P}(\cdot \mid x_1, a), \mathcal{P}(\cdot \mid x_2, a)\big)\Big), \tag{1}$$

$d^{\sim}(x_1', x_2')$ If P is deterministic

Specifically, the bisimulation relation is recovered as the zero-set of the metric:

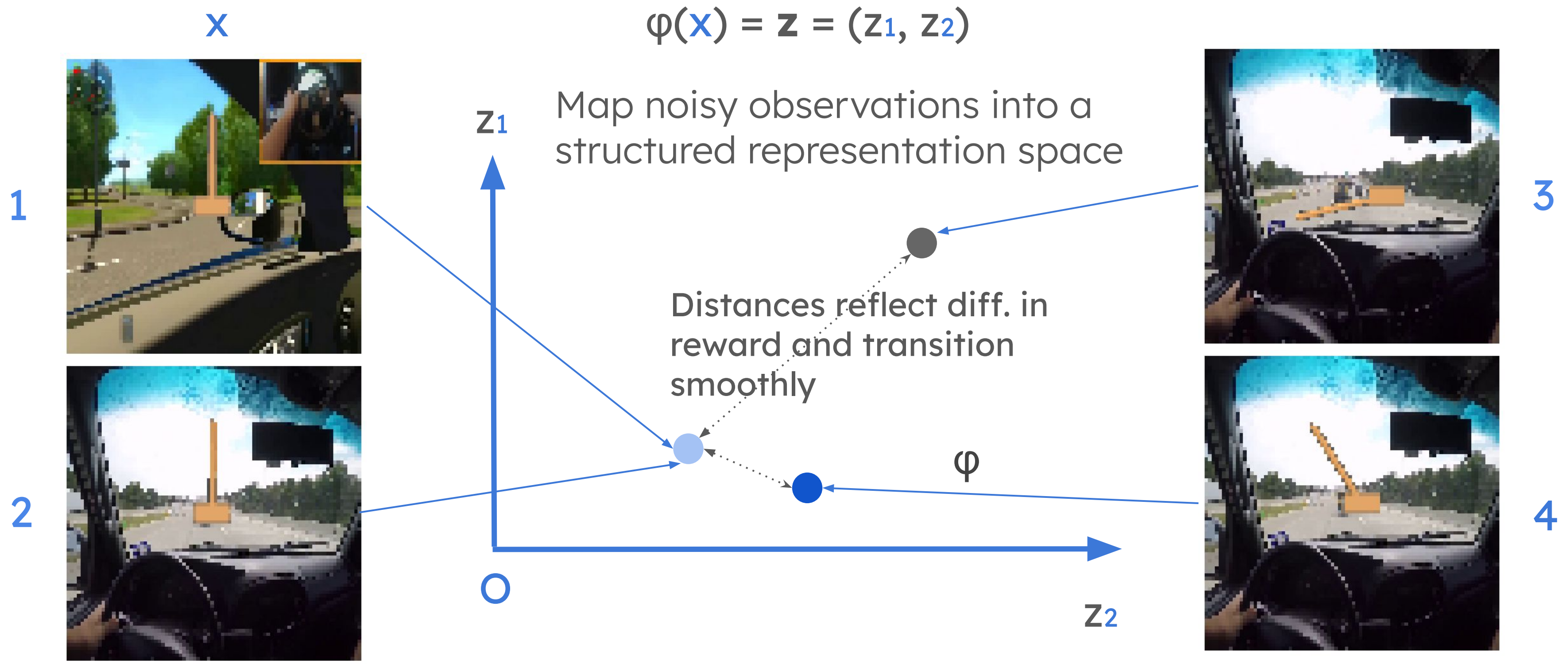
$$x_1 \sim x_2 \iff d^{\sim}(x_1, x_2) = 0.$$

Behavioral metrics: a broader class that quantify state similarity based on differences in R & P

Behavioral **Metric** (distance) Learning & Denoising **representation**

- **Behaviorally** *similar* states should have *close* representations, vice versa

Described by behavioral metric



Conceptual Analysis

We provide a unified framework, instantiating prior works.

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

Conceptual Analysis on Behavioral Metric Learning in RL

We aim to find an encoder that maps noisy observations into a structured representation space, which facilitates RL by ensuring that task-relevant variations are captured.
A natural way to formalize this goal is through an *isometric embedding*[1]:

Definition (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,

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

where $d_\mathcal{X}$ is the **target metric** and d_Ψ is the **representational metric**.

Target Metric $d_{\mathcal{X}}$: a **desired** behavioral distance between states

A target metric, inherent in an MDP, captures differences in rewards and transition dynamics:

$$d_{\mathcal{X}}(x_1, x_2) := c_R\, d_R(x_1, x_2) + c_T\, d_T(d_{\mathcal{X}})(\mathcal{P}(\cdot \mid x_1), \mathcal{P}(\cdot \mid x_2)),$$
$$\approx \hat{d}_{\mathcal{X}}(x_1, x_2) = c_R\, \hat{d}_R(r_1, r_2) + c_T\, \hat{d}_T(\hat{d}_{\mathcal{X}})(\hat{\mathcal{P}}(\cdot \mid x_1), \hat{\mathcal{P}}(\cdot \mid x_2)).$$

Here r_1, r_2 are sampled immediate rewards, d_R and d_T denote immediate and long-term similarity, and $\hat{d}_R$, $\hat{d}_T$ are their approximants.

Policy-dependent Bisimulation Metric (PBSM)

Recall BSM:

$$d^{\sim}(x_1, x_2) = \max_{a \in \mathcal{A}} \Big(c_R \, |\mathcal{R}(x_1, a) - \mathcal{R}(x_2, a)| + c_T \, \mathcal{W}_1(d^{\sim})\big(\mathcal{P}(\cdot \mid x_1, a), \mathcal{P}(\cdot \mid x_2, a)\big) \Big), \quad (1)$$

Policy-dependent bisimulation metrics (PBSMs) restrict similarity to the current policy, avoiding the max over actions. Define

$$\mathcal{R}^{\pi}(x) = \mathbb{E}_{a \sim \pi}[\mathcal{R}(x, a)], \quad \mathcal{P}^{\pi}(\cdot \mid x) = \mathbb{E}_{a \sim \pi}[\mathcal{P}(\cdot \mid x, a)].$$

Then

$$d^{\pi}(x_1, x_2) = c_R \, |\mathcal{R}^{\pi}(x_1) - \mathcal{R}^{\pi}(x_2)| + c_T \, \mathcal{W}_1(d^{\pi})\big(\mathcal{P}^{\pi}(\cdot \mid x_1), \mathcal{P}^{\pi}(\cdot \mid x_2)\big). \quad (2)$$

Matching under Independent Couplings (MICo)

MICo uses the independent coupling to approximate the 1-Wasserstein term, trading exactness for efficiency:

$$\begin{aligned} u^{\pi}(x_1, x_2) = c_R \left|\mathcal{R}^{\pi}(x_1) - \mathcal{R}^{\pi}(x_2)\right| \\ + \; c_T \, \mathbb{E}_{\substack{x_1' \sim \mathcal{P}^{\pi}(\cdot|x_1) \\ x_2' \sim \mathcal{P}^{\pi}(\cdot|x_2)}} \left[u^{\pi}(x_1', x_2')\right]. \end{aligned} \tag{3}$$

Here (x_1', x_2') are sampled independently from each transition.

Simple State Representation (SimSR)

Simple State Representation (SimSR) further replaces the true dynamics by a learned model $\widehat{\mathcal{P}}^{\pi}$ and embeds states isometrically using a **cosine distance**:

$$u^{\pi}(x_1, x_2) = c_R \left|\mathcal{R}^{\pi}(x_1) - \mathcal{R}^{\pi}(x_2)\right| + c_T \, \mathbb{E}_{\substack{x_1' \sim \widehat{\mathcal{P}}^{\pi}(\cdot|x_1) \\ x_2' \sim \widehat{\mathcal{P}}^{\pi}(\cdot|x_2)}} \left[u^{\pi}(x_1', x_2')\right]. \tag{4}$$

Under isometry,

$$u^{\pi}(x_1, x_2) = d_{\mathcal{X}}(x_1, x_2) = d_{\Psi}(\phi(x_1), \phi(x_2)) = 1 - \cos(\phi(x_1), \phi(x_2)).$$

Robust Approximation (RAP)

RAP (Chen and Pan, 2022) improves the approximation of the reward component d_R of the bisimulation metric by proposing a better surrogate $\hat{d}_R$. This is motivated by the following derivation:

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

- Here, r_x is a random variable such that $p(r_x = \mathcal{R}(x, a)) = \pi(a \mid x)$.

To approximate an isometric embedding: An example

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

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

$$\hat{d}_\mathcal{X}(x_1, x_2) = c_R \hat{d}_R(r_1, r_2) + c_T \hat{d}_T(d_\Psi)(\hat{\mathcal{P}}(\psi' \mid x_1), \hat{\mathcal{P}}(\psi' \mid x_2))$$

In DBC (Zhang et al., 2020), to approximate PBSM (Def. 6), the metric loss is defined in the following form:

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

Other Important Design choices in Metric Learning

- Self-prediction (**ZP**) loss $J_{\text{ZP}}(\phi, \nu) = -\log P_\nu(\bar{\phi}(x') \mid \phi(x), a),$
- Reward prediction (**RP**) loss $J_{\text{RP}}(\phi, \kappa) = (R_\kappa(\phi(x), a) - r)^2,$
- Metric loss function *l* (MSE/Huber) $J_M(\phi) = \boxed{\ell}\left(d_\Psi(\phi(x_1), \phi(x_2)) - \hat{d}_{\mathcal{X}}(x_1, x_2)\right),$
- Target trick - using target network for one observation in d_Ψ:

$$U_\omega(x, y) = \frac{\|\phi_\omega(x)\|_2^2 + \|\phi_{\boxed{\bar{\omega}}}(y)\|_2^2}{2} + \beta\theta(\phi_\omega(x), \phi_{\bar{\omega}}(y))$$

(from MICo)

Other Important Design choices in Metric Learning

Normalization in the representation space

- L2 normalization $$\text{L2Norm}(\psi) = \frac{\psi}{\|\psi\|_2}.$$ (from SimSR)
- MaxNorm: adjust the diameter of the vector so that d_Ψ is bounded theoretically

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

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

(from DBC-normed)

- LayerNorm (as default design choice in CNNs in prior work)

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

Two baselines, five metric learning methods

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

Conceptual Analysis: Denoising and Metric Learning

Many works motivate metric learning through denoising. But,

- What is denoising exactly?
- Why (why not) metrics denoise?

This motivates our study design.

BMDP, EX-BMDP: Formalizing Distracting Environments

A block MDP (Du et al., 2019) is a tuple

$$\langle \mathcal{X}, \mathcal{Z}, \mathcal{A}, q, p, R, \gamma \rangle,$$

where

(underlying)

$\mathcal{X}$: observation space, $\mathcal{Z}$: latent state space, $\mathcal{A}$: action space,

$q \colon \mathcal{Z} \to \Delta(\mathcal{X}), \quad x \sim q(\cdot \mid z),$

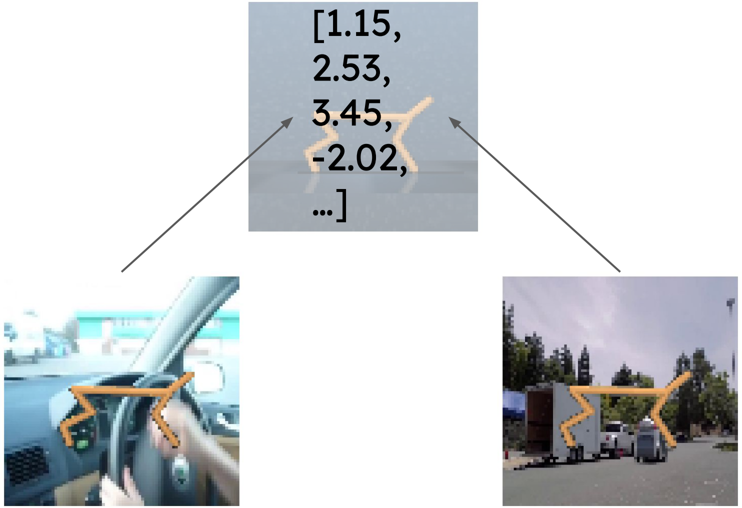
$p \colon \mathcal{Z} \times \mathcal{A} \to \Delta(\mathcal{Z}), \quad R \colon \mathcal{Z} \times \mathcal{A} \to \mathbb{R}, \quad \gamma \in [0, 1).$

Block structure:

$$\forall z_1, z_2 \in \mathcal{Z},\ z_1 \neq z_2 \implies \operatorname{supp}(q(\cdot \mid z_1)) \cap \operatorname{supp}(q(\cdot \mid z_2)) = \emptyset,$$

guarantees existence of an oracle encoder (inverse) $q^{-1} : \mathcal{X} \to \mathcal{Z}$.

- One z can correspond to many x
- Each x corresponds to only one z
- Exist an oracle encoder that recovers z from x



An EX-BMDP (Efroni et al., 2021) extends the block MDP by decomposing

$$\mathcal{Z} = \mathcal{S} \times \Xi, \quad z = (s, \xi),$$

where $s \in \mathcal{S}$ is the task-relevant state and $\xi \in \Xi$ is exogenous noise. Transitions factorize as

$$p(s', \xi' \mid s, \xi, a) = p(s' \mid s, a) \; p(\xi' \mid \xi),$$

The reward is independent of noise:

$$R \colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}.$$

EX-BMDPs guarantee a denoising map

$$D \colon \mathcal{Z} \to \mathcal{S},$$

and combined with the oracle encoder q^{-1} one recovers

$$\phi^*(x) = D(q^{-1}(x)), \quad s_t = \phi^*(x_t).$$

s: robot state
ξ: video frame index
q: a rendering function

What is denoising?

We define denoising as the removal of task-irrelevant noise ξ. Formally:

Definition (Perfect Denoising)

An encoder ϕ achieves **perfect denoising** in an EX-BMDP if, for any triplet $x, x_+, x_- \in \mathcal{X}$ satisfying

$$\phi^*(x) = \phi^*(x_+) \neq \phi^*(x_-),$$

it holds that

$$\phi(x) = \phi(x_+) \neq \phi(x_-).$$

That is, ϕ exactly replicates the abstraction of the oracle encoder ϕ^*.

Why do Metrics Help with Denoising?

- **Bisimulation metric (BSM)**: Achieves perfect denoising in EX-BMDP ($d_{\mathcal{X}}(x, x_+) = 0$), so isometric embedding maps bisimilar observations to identical representations (Ferns et al., 2004, 2011).
- **Policy-dependent Bisimulation (PBSM)**: Guarantees denoising when the policy is *exo-free* (Islam et al., 2022).
- **MICo distance**: Does not generally assign zero distance to bisimilar pairs unless both policy and transitions are deterministic, yet empirical evidence shows it can still cluster behaviorally similar observations (Castro et al., 2021; Chen and Pan, 2022; Zang et al., 2022).

Why do Metrics not Help with Denoising?

- **Intractable BSM**: Exact computation is prohibitive, leading to reliance on PBSM and MICo approximants (Castro, 2020).
- **Policy-dependence of PBSM**: May fail to denoise under arbitrary (even optimal) policies (Islam et al., 2022).
- **Off-policy sampling**: Approximated reward metric $\hat{d}_R$ uses replay-buffer data, conflicting with the on-policy metric assumption.
- **Model approximation error**: Learned transition models introduce bias in $\hat{d}_T$ (Kemertas and Aumentado-Armstrong, 2021).
- **Loss interactions**: Metric loss combined with ZP and critic losses can degrade denoising effectiveness in practice.

Recap

1. **State abstraction**: aggregation, bisimulation relation, metrics
2. **Isometric embedding:** connecting metric and representation learning
3. A general form of **target metrics** (in benchmarked works): [$d_R + d_T$]
4. How to **approximate** the target metrics
5. Promising application: denoising [removing task-irrelevant noise]
6. Why / why not metrics help with denoising?

Our Study Design

Driven by multiple research questions, we think critically about how to move this area forward.

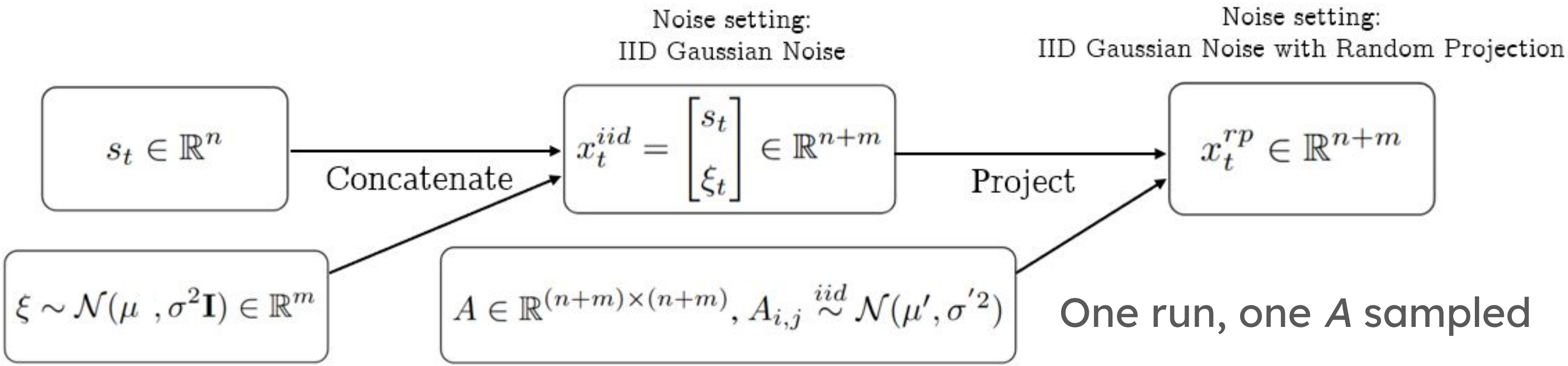
Aspect	Prior Work	Our Study Design
Task Diversity	Limited test environments: few tasks with *grayscale natural video* backgrounds	Diverse **state-based and pixel-based** noise settings across tasks
Generalization Evaluation	Entangled: evaluation only on *unseen* videos (OOD), hard to know the source of difficulty	Clear separation of **ID and OOD generalization** via distinct train/test noise
Evaluation Signal	Indirect: impact on *evaluation return*	Direct: proposed **Denoising Factor (DF)** as a targeted representation measure
Loss Design	Mixed: multiple intertwined losses obscure metric learning effect	**Isolated metric evaluation** disentangles representation from RL objectives

Noise Settings

★ Introduce diverse **state-based** and **pixel-based** noise settings based on EX-BMDP

State-based envs:

- IID Gaussian Noise (dims and stds can be varied)
- IID Gaussian Noise with *Random Projection*

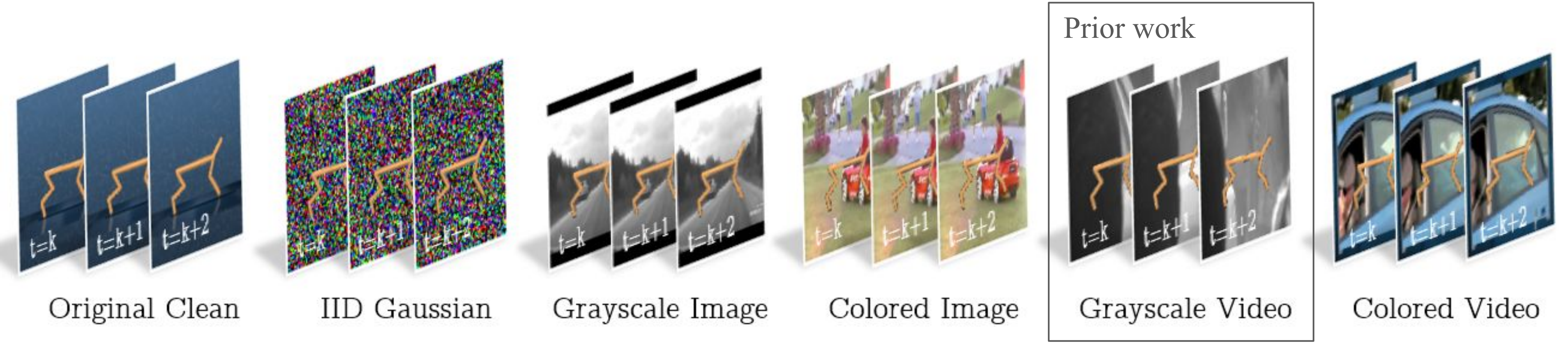


One run, one *A* sampled

Noise Settings

Pixel-based envs (backgrounds can be grayscale or colored):

- **IID Gaussian Noise** applied per-pixel
- **Natural Images**: replacing clean background with **one** randomly selected image (consistent in a run) -> visual complexity only
- **Natural videos**: replacing clean background with videos (playing in a loop); temporally dependent

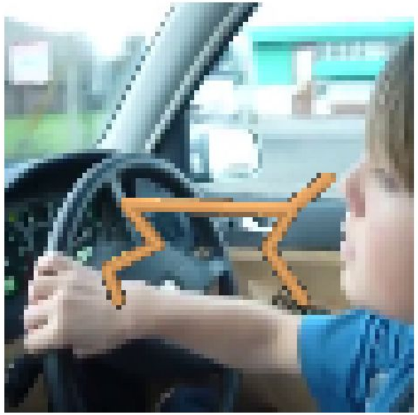


In-distribution (ID) vs. Out-of-distribution (OOD) Generalization

The training and testing environment are **identical**

Training

Evaluation

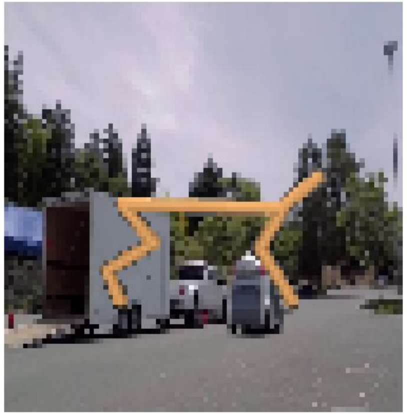


ID Generalization

The training and testing environment share the **same task-relevant parts** but **differ in noise distributions**

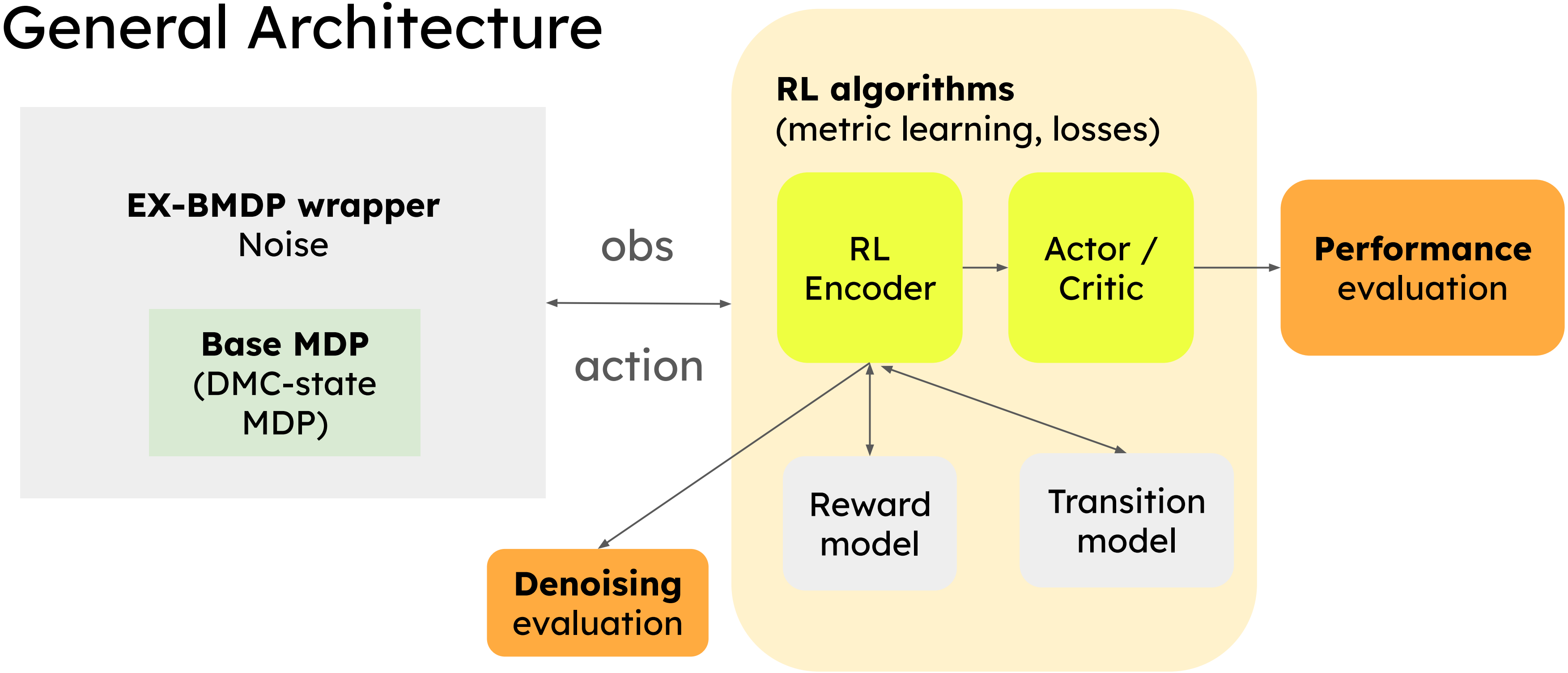
Training

Evaluation



OOD Generalization (prior work)

General Architecture



Quantifying Denoising

We introduce the *denoising factor* (DF), a measure that **quantifies** an encoder's ability to **denoise**.

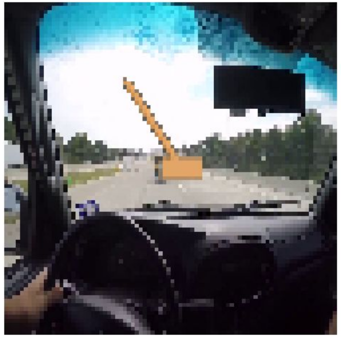
Positive examples x+

Agent can view them
as same observations

Anchor x

Negative examples x-

Any randomly sampled
observations



Positive & Negative Scores

To compute the denoising factor, select an anchor $x \sim \rho_\pi$, a positive example x_+ with $\phi^*(x_+) = \phi^*(x)$, and a negative example x_- sampled IID.

Definition (Positive score)

$$\mathrm{Pos}^{\pi}_{d_\Psi}(\phi) \;:=\; \mathbb{E}_{\substack{x \sim \rho_\pi,\, \xi_+ \sim \rho(\xi_+), \\ x_+ \sim q(\cdot|\phi^*(x), \xi_+)}} \left[d_\Psi\left(\phi(x), \phi(x_+)\right)\right].$$

Definition (Negative score)

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

Denoising Factor (DF)

The denoising factor measures the normalized difference between negative and positive scores:

Definition (Denoising factor)

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

A higher DF indicates stronger denoising ability, with the oracle encoder ϕ^* achieving $\mathrm{DF} = 1$.

Isolated Metric Estimation Setting

Agent encoder:

- Optimized by **RL losses** (e.g., Q loss) and used in end-to-end training

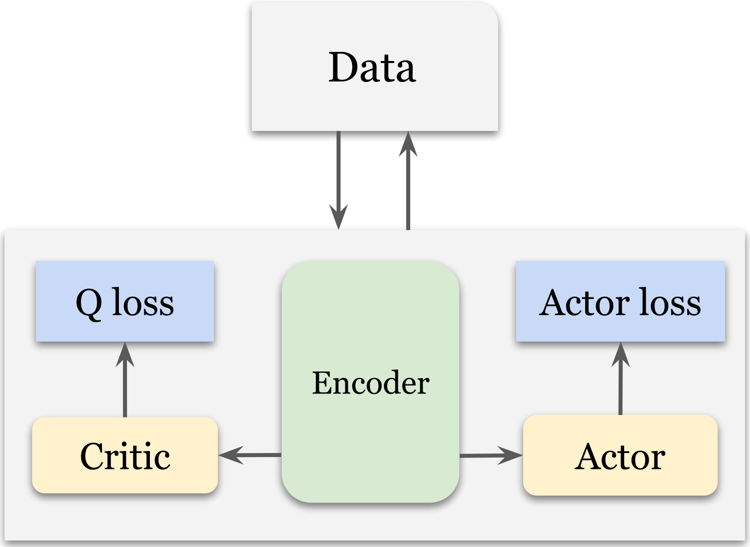
Isolated metric encoder:

- Optimized by **metric losses** (or more broadly, a different combination of objectives than agent encoder),
- Only used to **evaluate DF**

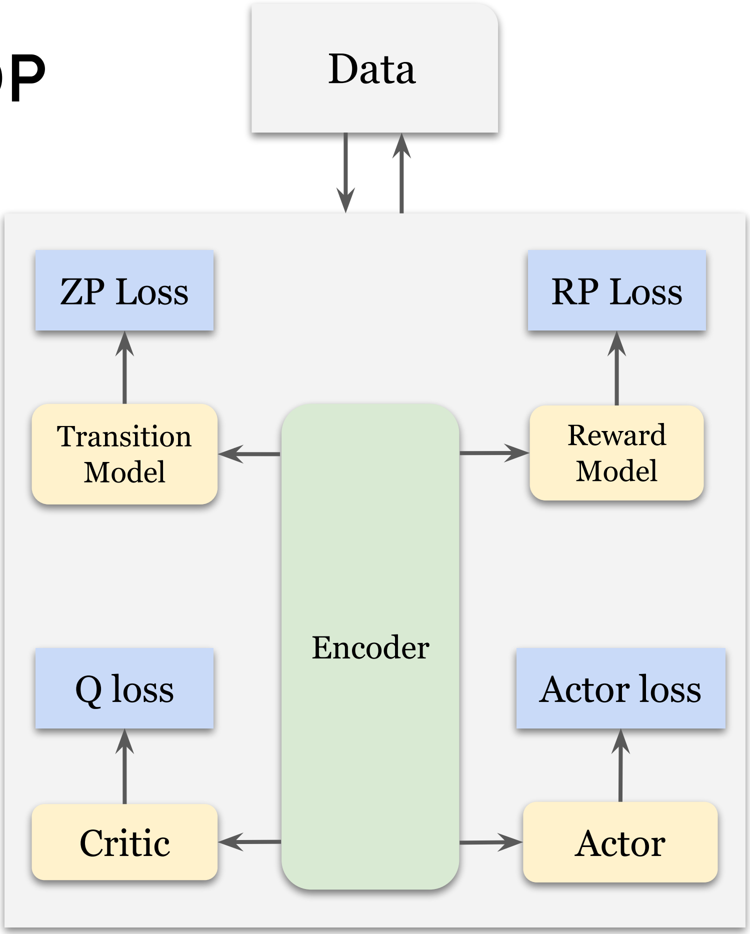
√ Remove other losses on representation from analysis

√ Ensure a fixed data collection (π), and enable fair comparison of denoising capability (DFs) of different isolated metric encoders!

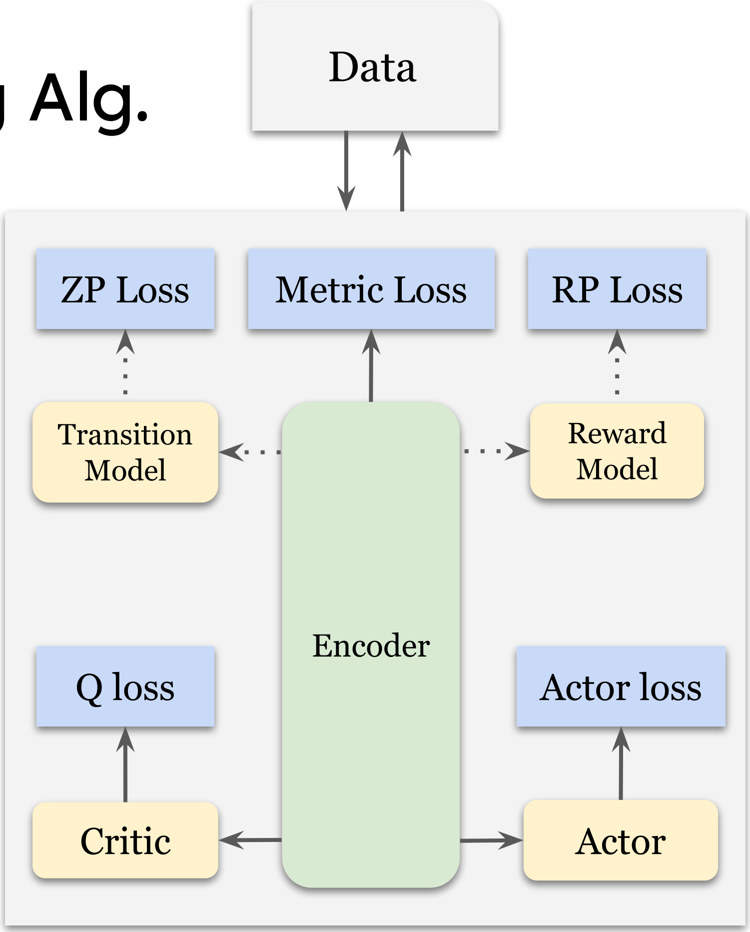
SAC Architecture



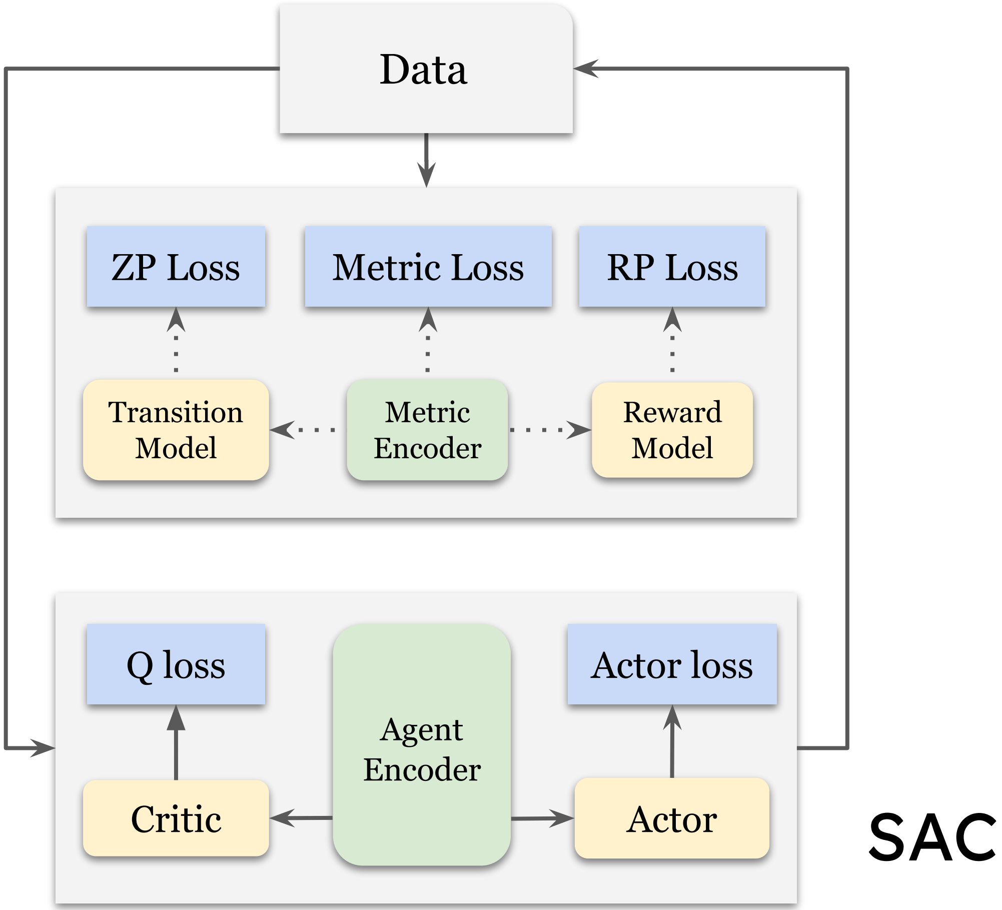
DeepMDP



Metric Learning Alg.



Isolated Metric Evaluation
(an instantiation)



Experiment

- **Benchmarking** result on various tasks and noise settings
  - Understanding overall task difficulty and agent's performance on aggregate (~300 settings)
- **Case study**: What matters in metric (and representation) learning?
  - Identifying key design choices that lead to performance gain
- **Isolated Metric Evaluation Setting:** Does Metric Learning Help with Denoising?
  - Understanding the connection between metric learning and denoising
- **OOD Generalization Evaluation** on Pixel-based Tasks
  - The setting of interest in previous work

Comprehensive Benchmarking

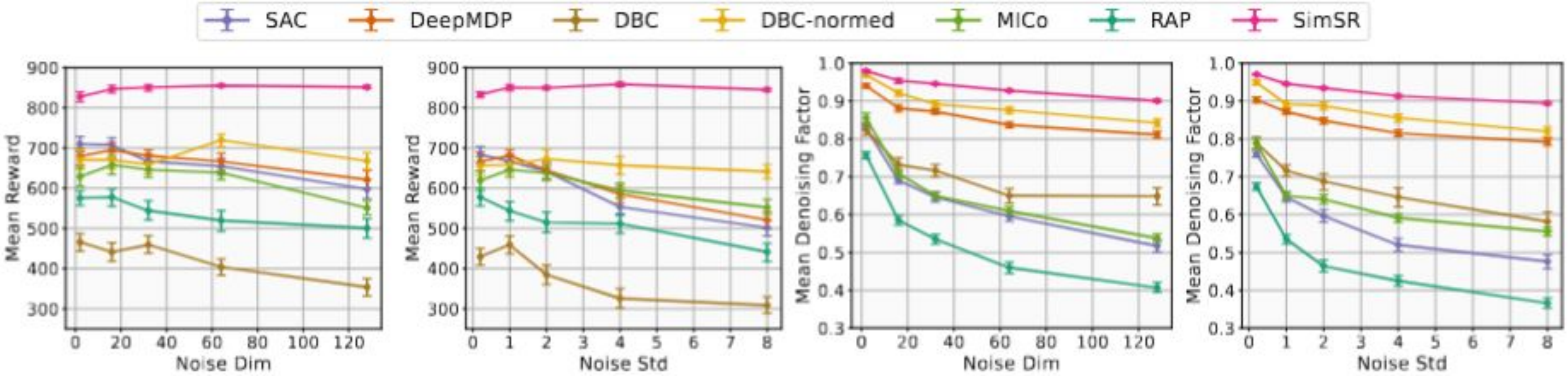
Settings (in DMC):

- 20 state-based tasks * 10 IID Gaussian noises (varying dim/std)
- 14 pixel-based tasks * 6 background noises

Aggregating result respectively across:

- All tasks, in benchmarking section
- 12 seeds for state-based, 5 seeds for pixel-based envs
- Each run we aggregate 10 eval point from 1.95M-2.05M

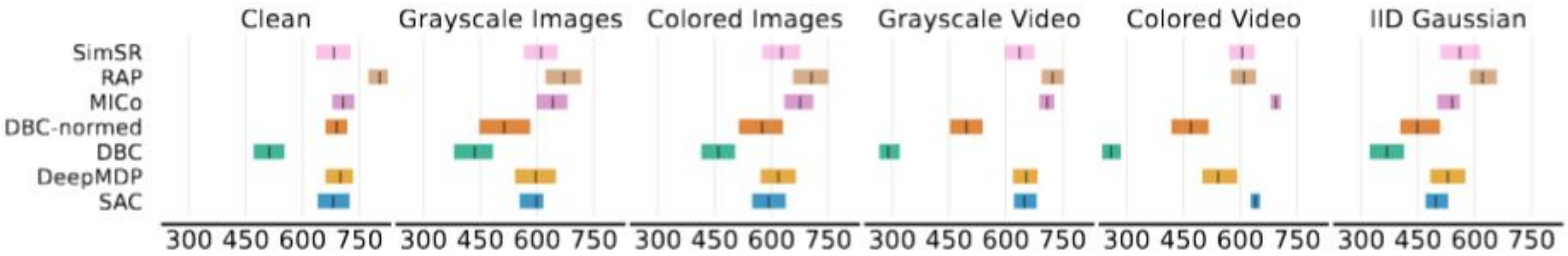
State-based benchmarking result: IID Gaussian Noise



- SimSR perform the best (but why?)
- Increasing the number of noise dimensions cause moderate reward drop
- Well-performing methods are robust to noise variations

Pixel-based benchmarking result

- RAP generally perform the best (but not in state-based tasks!)
- Grayscale video setting is *not much harder* than clean background setting!



Additional objectives trade off computation efficiency

Relative update time

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

- Optimizing a metric loss (e.g., in MICo) is as expensive as a ZP loss (e.g., in DeepMDP), per runtime comparison.

(show state-based as example)

Task difficulty benchmarking:

Aggregating scores for different agents (7 methods, all noise settings)

A wide spectrum of task difficulty!

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

Case study: with (R') / w/o (R) LayerNorm on representation

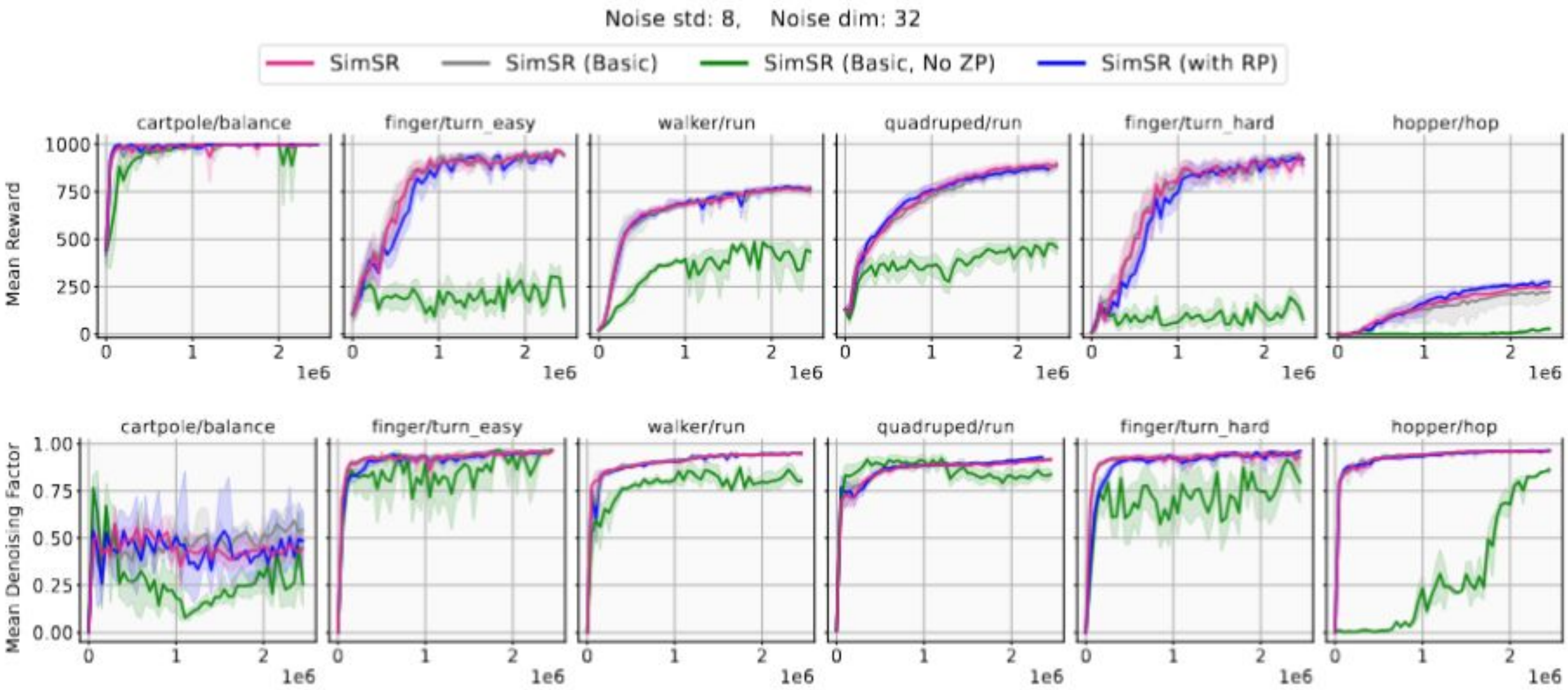
6 representative easy-to-hard tasks are sampled for later analysis.

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

Most methods benefit from LayerNorm in the representation space

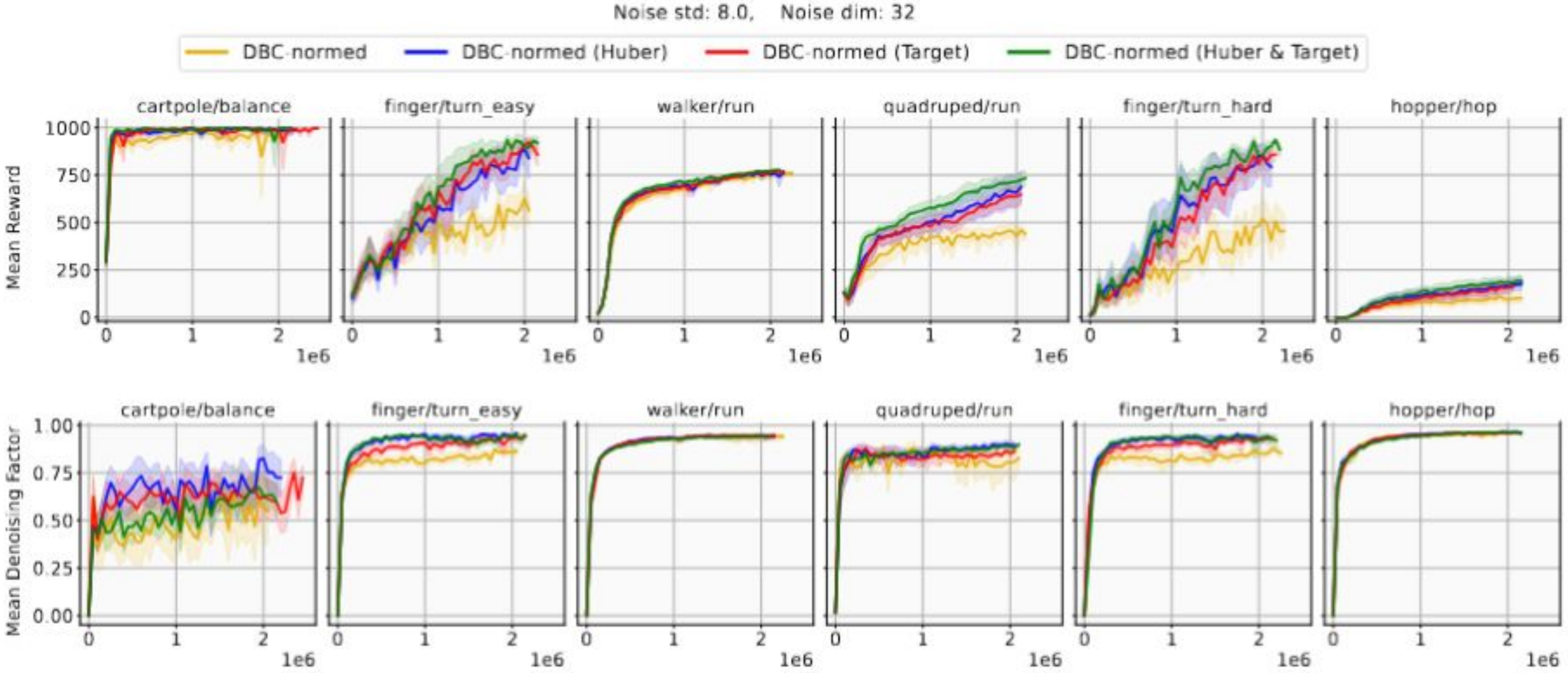
DeepMDP (RP+ZP) with LayerNorm performs comparably to SimSR

Case study: SimSR with / w/o ZP



- ZP is essential to SimSR's success!

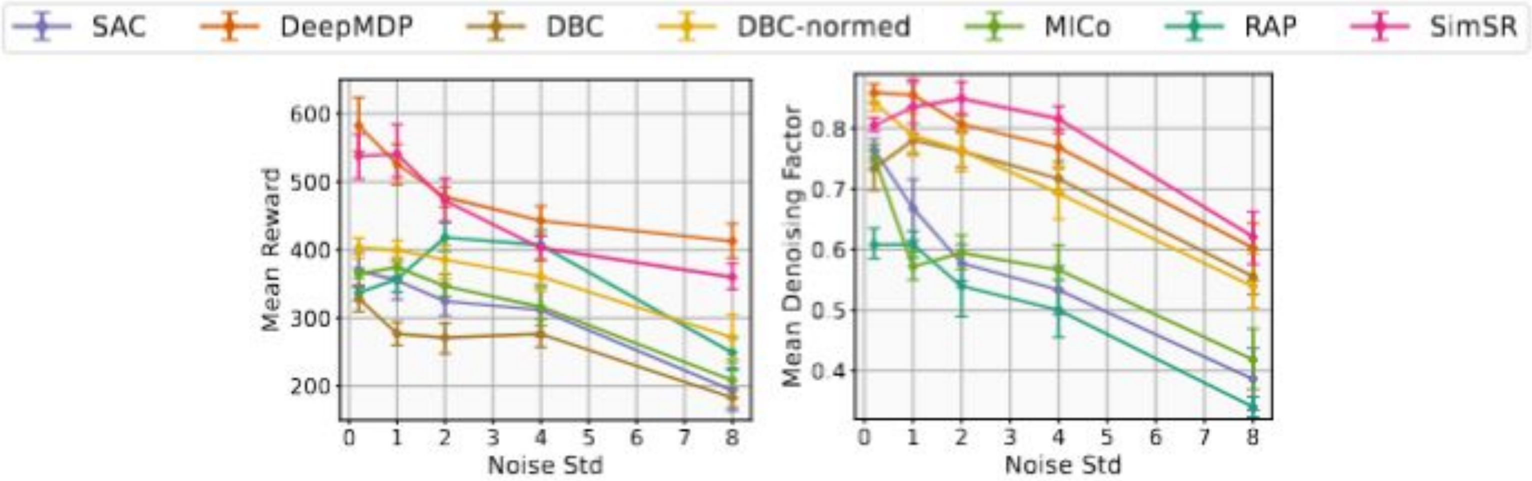
Case study: other design choices



- Huber loss and target trick provides moderate amount of help

Case study: hard noise setting (IID Gaussian with **random projection**)

6 representative tasks aggregation



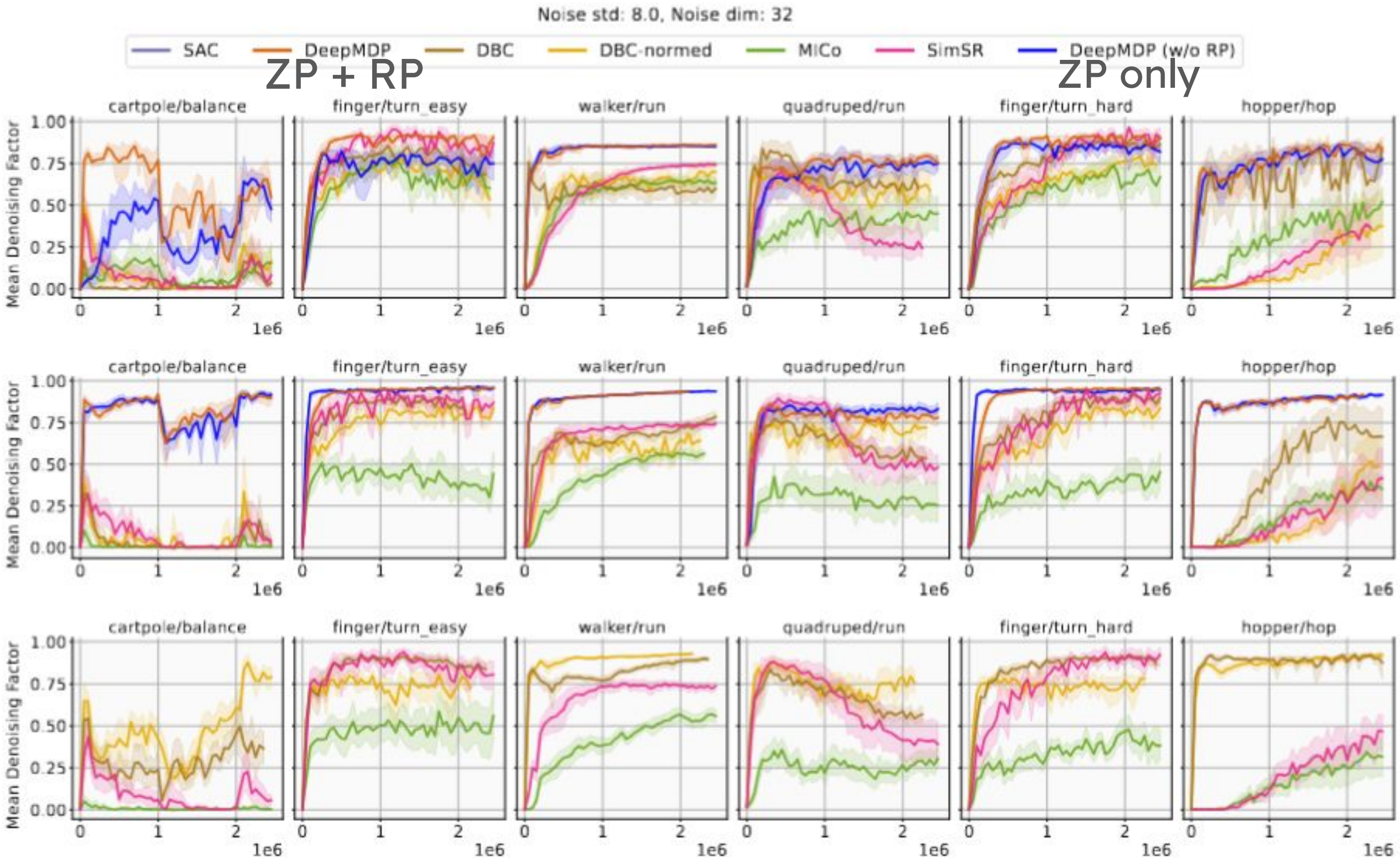
Much harder, but DeepMDP / SimSR remain relatively robust

Case study: takeaways

- **Most methods benefit from LayerNorm** in the representation space
    - may due to a stable representation and gradient norms, and help numerical stability in extrapolation of the metrics and Q values
- DeepMDP with LayerNorm performs comparably to SimSR
- **ZP loss is crucial for SimSR's success** in noisy state-based task (though many methods even do not show they are using ZP!)
    - DeepMDP (RP+ZP) + LayerNorm on par with SimSR
- Other tricks help but marginal
- DeepMDP and SimSR remain relatively robust to the hard IID Gaussian + random projection noise

Isolated Metric Evaluation Setting

Learned metric denoises, but not better than the representation obtained by optimizing ZP



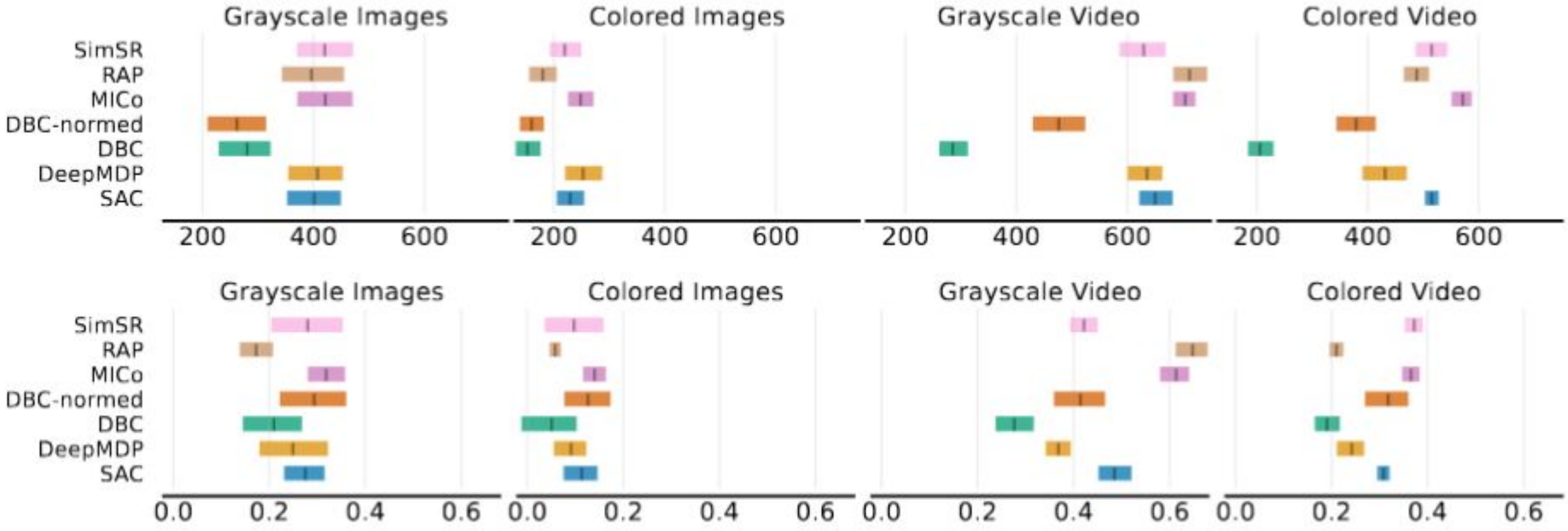
No LayerNorm

With LayerNorm

ZP + metric loss

OOD Generalization for 14 pixel-based tasks

Methods struggle to generalize in both grayscale and colored image settings (lacking of “domain randomization”)



Grayscale video noise (widely used) is not challenging enough for OOD generalization as baselines generalize well.

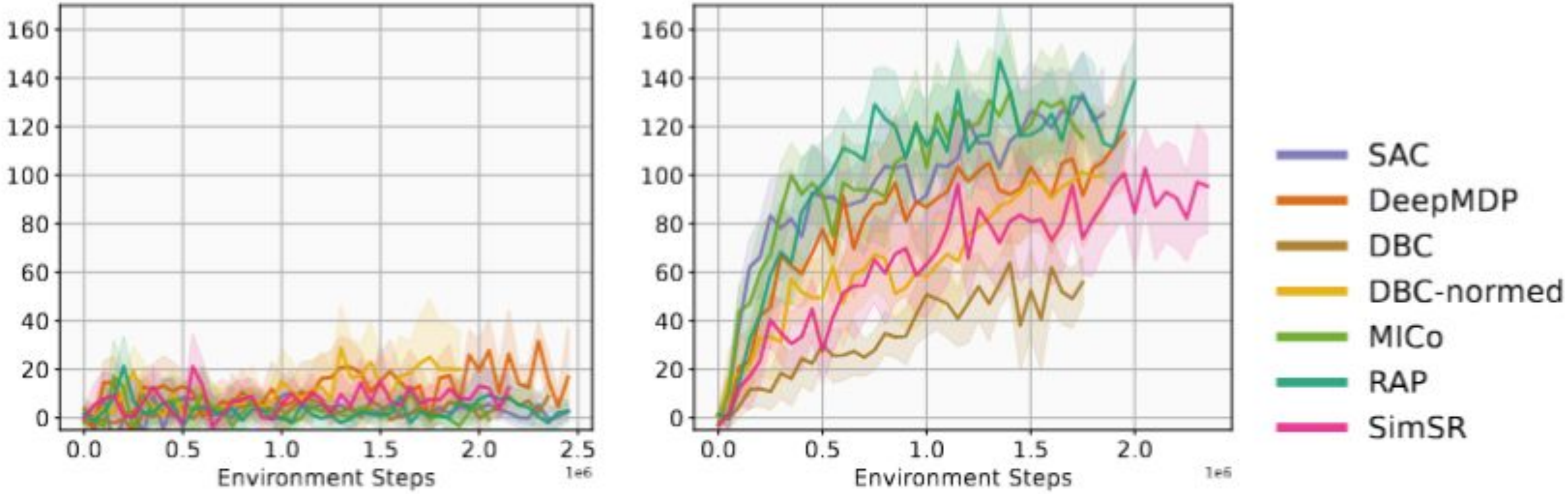


Figure 10: **Reward gap** (performance in ID evaluation minus OOD evaluation) in the grayscale video setting (left) and the colored video setting (right), aggregated on 14 pixel-based tasks in Table 9.

Thanks for your attention!

Takeaways

Blog

Paper

- Evaluate first in **simple, controlled settings** to build foundational insight
- Support metric-learning claims via **direct measures** (e.g, denoising factor) and distinguishing ID vs. OOD generalization
- **Self-prediction (ZP) loss** and **normalization** schemes are decisive design choices shaping representation and metric quality
- Examine when metric learning offers **unique benefit**, since incorporating ZP loss and LayerNorm into SAC can achieve similar advantages

Knowledge Map