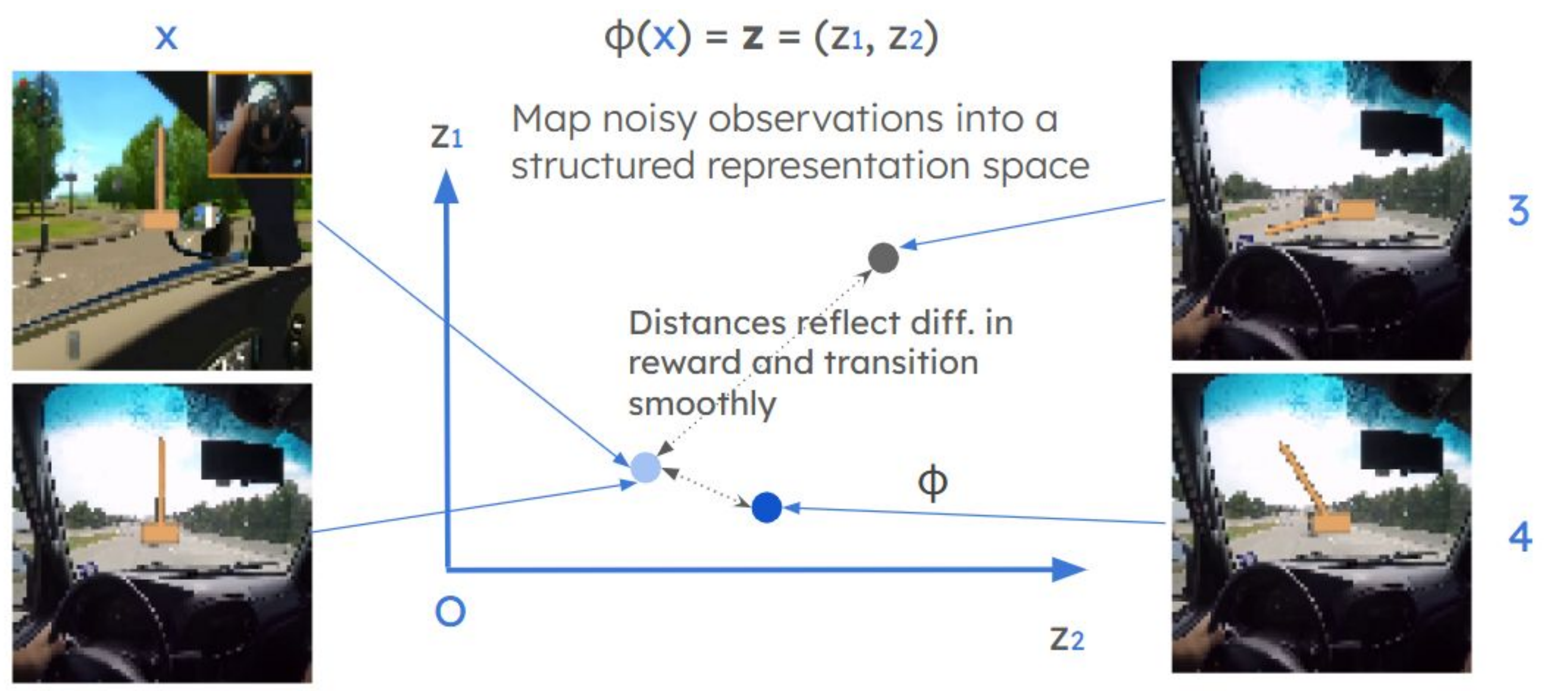
Understanding Metric Learning

A Large-Scale Study on Distracting RL Environments

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

WHAT & WHY Metric Learning

- Metric learning ≈ learning distances between states by how they behave (rewards + transitions)



- Principled way for state abstraction: Pull behaviorally similar states together, Push dissimilar ones apart
- Embed a *target distance* isometrically to repr. space

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

- Bisimulation metric (BSM) is the canonical *target distance*

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

- Inspired by BSM, scalable variants:

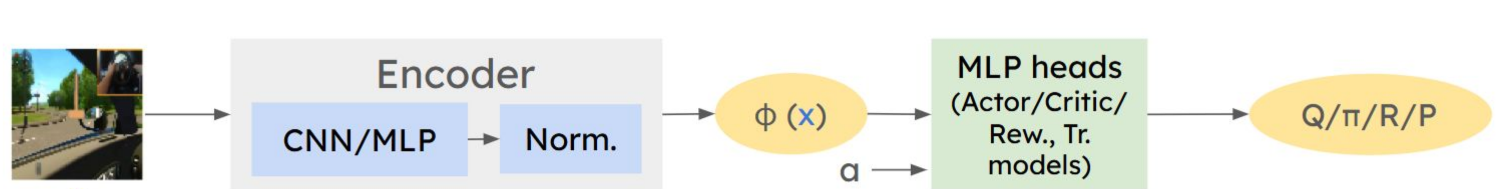
$$d^\pi(x_1,x_2) = \underbrace{c_R|\mathcal{R}^\pi(x_1) - \mathcal{R}^\pi(x_2)|}_{d_R} + \underbrace{c_T\,\mathcal{W}_1(d^\pi)(\mathcal{P}^\pi(x_1), \mathcal{P}^\pi(x_2))}_{d_T}.$$ Policy-dependent BSM

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

- Approximate isometry through a regression loss

$$J_M(\phi) = \ell\left(d_\Psi(\phi(x_1), \phi(x_2)) - \hat{d}_{\mathcal{X}}(x_1,x_2)\right) \qquad \ell: \text{MSE/Huber/…}$$

- Important design choices



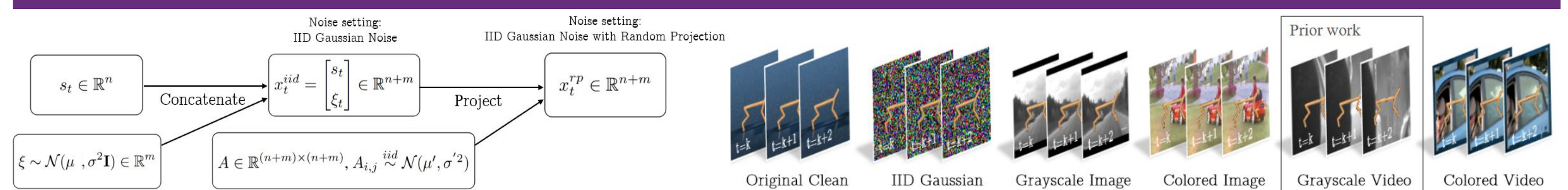
- Objectives:
  - Reward Prediction (RP) $J_{\mathrm{RP}}(\phi,\kappa) = (R_\kappa(\phi(x), a) - r)^2$
  - Self-prediction (ZP) $J_{\mathrm{ZP}}(\phi,\nu) = -\log P_\nu(\bar{\phi}(x') \mid \phi(x), a)$
  - Metric Loss $J_M(\phi) = \ell\left(d_\Psi(\phi(x_1), \phi(x_2)) - \hat{d}_{\mathcal{X}}(x_1,x_2)\right)$
- Normalizing Representations

L2Norm | LayerNorm | MaxNorm

"How Well Are Metrics Learned in Deep RL?"

Paper   Github   Blog

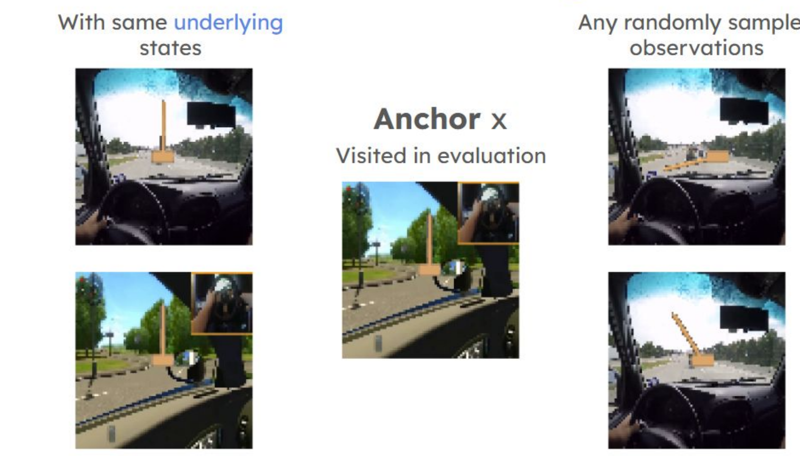
Extensive Noise Sources



State-based
- IID Gaussian (append)
- IID Gaussian with random projection

Pixel-based
- Static image (gray / colored)
- Video background (gray / colored)
- IID Gaussian (per-pixel)

Quantifying Denoising



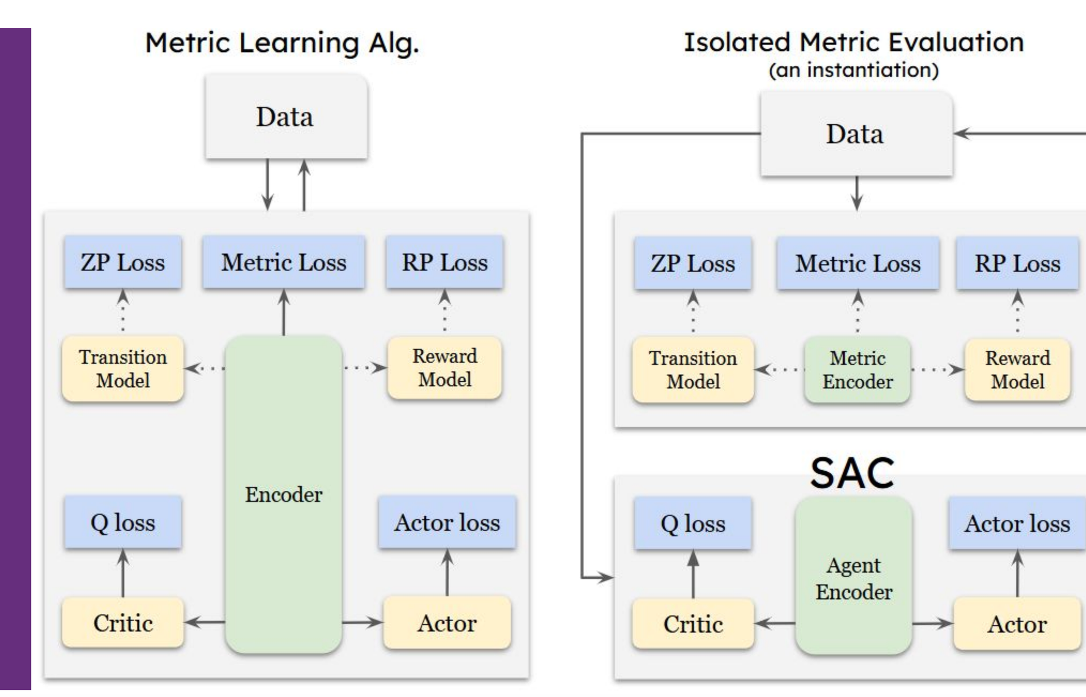
$$\text{Pos} := \mathbb{E}[d_\Psi(\phi(\;), \phi(\;))]$$
Positive examples x+

$$\text{Neg} := \mathbb{E}[d_\Psi(\phi(\;), \phi(\;))]$$
Negative examples x+

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

Isolate Metrics Effect

- A baseline agent + An isolated encoder
  - optimized by only metric-related loss
  - only evaluate its DF
- **Disentangle** other losses from the encoder
- **Same π** (from SAC) for different metric losses



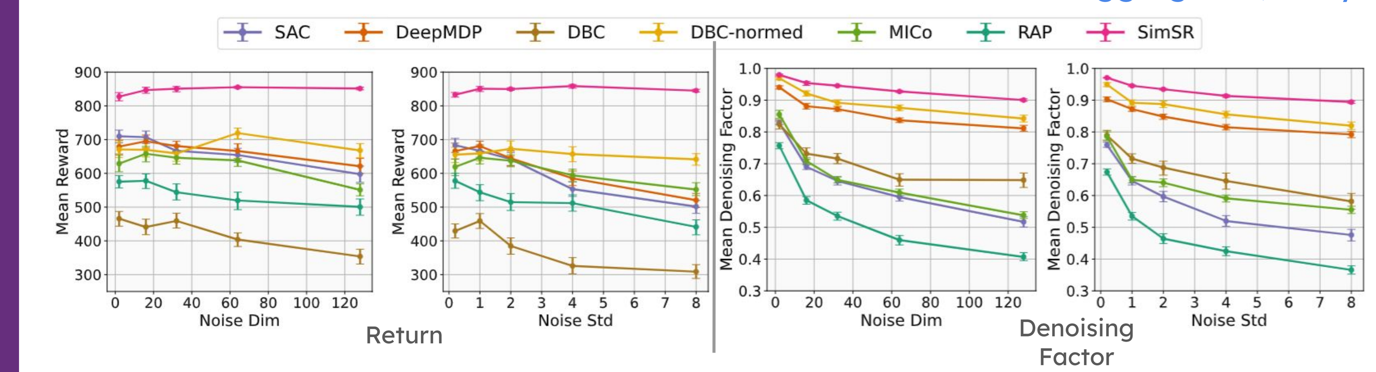
Unified Perspective

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

Mila   McGill   Université de Montréal   University of Toronto

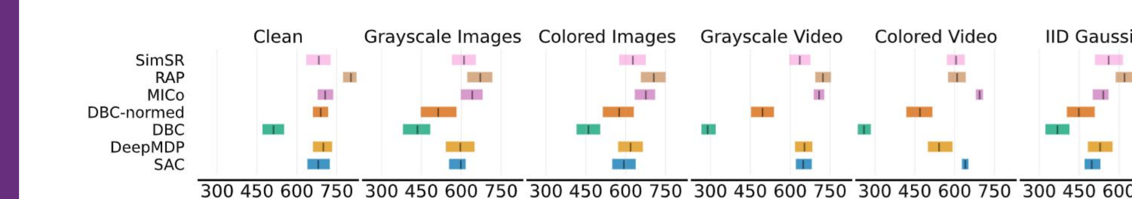
BENCHMARKING

20 DMC tasks aggregated, - LayerNorm



- **SimSR** perform the best (but why?)
- Increasing noise dim/std -> moderate reward drop
- Well-performing methods are robust to noise variations

14 DMC tasks aggregated, + LayerNorm



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

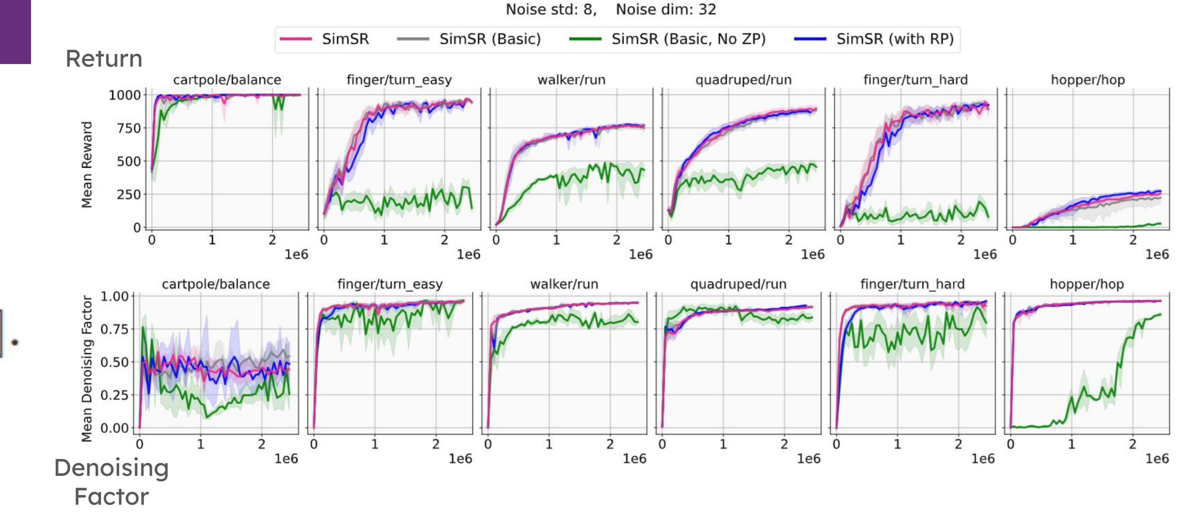
CASE STUDY

Task	Return	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

Red: R' > R   Blue: R' < R

Most methods benefit from LayerNorm in the repr. space

DeepMDP (RP+ZP) + LayerNorm ≈ SimSR

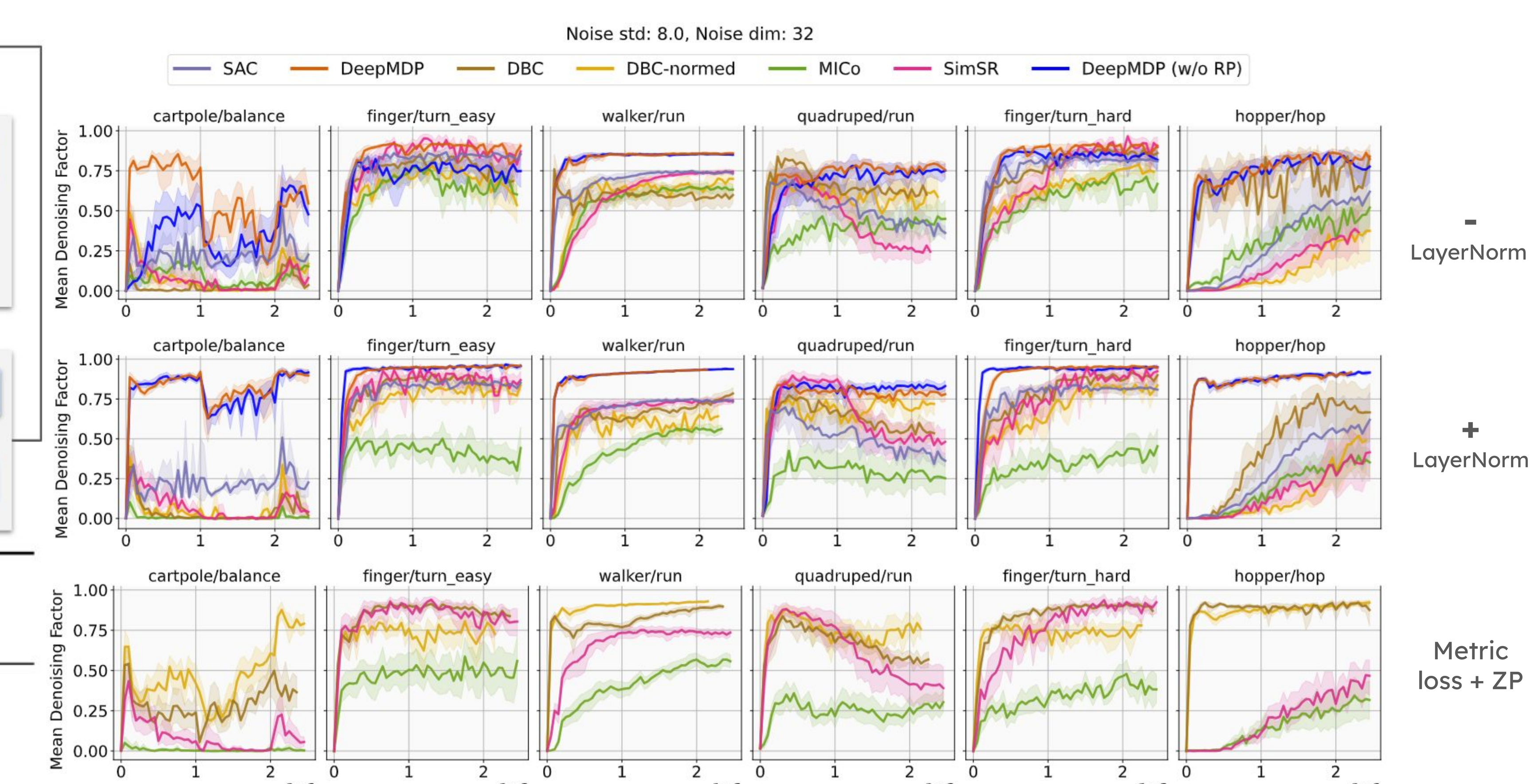


ZP is essential for the success of SimSR

RP does not matter too much in our tasks

And more...

ISOLATED EVALUATION



Learned metric denoises, but not better than by optimizing ZP (with LayerNorm)

TAKEAWAYS

★ Evaluate **simple, controlled settings first** to build foundational insight
★ Support metric-learning claims via **direct measure**
★ **Self-prediction (ZP) loss** & **Normalization** truly matters
★ Examine when metric learning offers **unique benefit**