Skill-aware Mutual Information Optimisation for Generalisation in Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Meta-Reinforcement Learning (Meta-RL) agents can struggle to operate across tasks with varying environmental features that require different optimal *skills* (i.e., different modes of behaviours). Using context encoders based on contrastive learning to enhance the generalisability of Meta-RL agents is now widely studied but faces challenges such as the requirement for a large sample size, also referred to as the $\log$-K curse. To improve RL generalisation to different tasks, we first introduce **S**kill-**a**ware **M**utual **I**nformation (SaMI), an optimisation objective that aids in distinguishing context embeddings according to skills, thereby equipping RL agents with the ability to identify and execute different skills across tasks. We then propose **S**kill-**a**ware **N**oise **C**ontrastive **E**stimation (SaNCE), a K-sample estimator used to optimise the SaMI objective. We provide a framework for equipping an RL agent with SaNCE in practice and conduct experimental validation on modified MuJoCo and Panda-gym benchmarks. We empirically find that RL agents that learn by maximising SaMI achieve substantially improved zero-shot generalisation to unseen tasks. Additionally, the context encoder equipped with SaNCE demonstrates greater robustness to reductions in the number of available samples, thus possessing the potential to overcome the $\log$-K curse.

1 Introduction

Meta-Reinforcement Learning (Meta-RL) agents often learn policies that do not generalise across tasks in which the environmental features and optimal *skills* are different [des Combes et al., 2018, Garcin et al., 2024]. Consider a set of cube-moving tasks where an agent must move a cube to a goal position on a table (Figure 1). These tasks become challenging if environmental features, such as table friction, vary between tasks. When facing an unknown environment, the agent needs to explore effectively, understand the environment, and adjust its behaviour accordingly within an episode. For instance, if the agent tries to push a cube across a table covered by a tablecloth and finds it "unpushable," it should infer that the table friction is relatively high and adapt by lifting the cube to

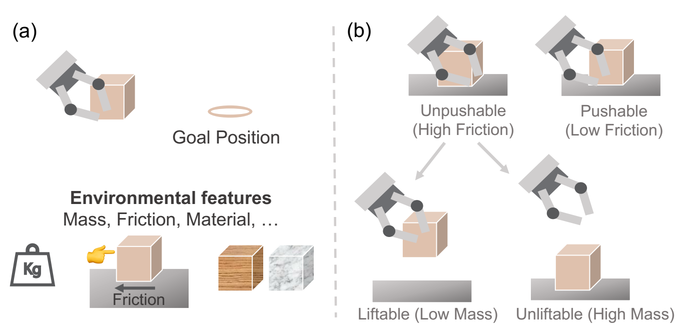


Figure 1: (a) In a cube-moving environment, tasks are defined according to different environmental features. (b) Different tasks have different transition dynamics caused by underlying environmental features, hence optimal skills are different across tasks.

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

avoid friction, rather than continuing to push. Recent advances in Meta-RL [Lee et al., 2020, Agarwal et al., 2021, Mu et al., 2022, Dunion et al., 2023b,a] have shown promising potential to perceive and understand environmental features by inferring context embeddings from a small number of interactions in the environment. The context embedding is expected to capture the distribution of tasks and efficiently infer new tasks. Meta-RL methods then train a policy conditioned on the context embedding to generalise to multiple tasks.

The context encoder is the key component for capturing the context embedding from recent experiences [Clavera et al., 2019a, Lee et al., 2020], which affects generalisation performance significantly. Some Meta-RL algorithms [Fu et al., 2021, Wang et al., 2021, Li et al., 2021, Sang et al., 2022] are equipped with context encoders based on contrastive learning, which uses the InfoNCE lower bound [Oord et al., 2019] to optimise the mutual information (MI) between context embeddings and trajectories. InfoNCE is a K-sample MI estimator and is utilised to learn distinct context embeddings for each task. Despite significant advancements, integrating contrastive learning with Meta-RL poses several unresolved challenges, of which two are particularly relevant to this research: **(i) existing context encoders based on contrastive learning do not distinguish tasks that require different skills**; many prior algorithms only pull embeddings of the same tasks together and push those of different tasks apart. However, for example, a series of cube-moving tasks with high friction may only require a Pick&Place skill (picking the cube off the table and placing it at the goal position), making further differentiation unnecessary. **(ii) *K*-sample MI estimators are sensitive to the sample size *K* (i.e., the *log-K* curse)** [Poole et al., 2019]; a substantial quantity of negative samples is required to approximate the true MI, which is challenging due to RL's low sample efficiency [Franke et al., 2021] and often impractical for achieving accurate MI estimation [Arora et al., 2019, Nozawa and Sato, 2021]. The effectiveness of K-sample MI estimators breaks down with a small sample size and leads to a significant performance drop in downstream RL tasks [Mnih and Teh, 2012, Guo et al., 2022].

To enhance RL generalisation across different tasks, we propose that the context embeddings should optimise downstream tasks and indicate whether the current skill remains optimal or requires further exploration, thereby addressing issue (i). This approach also reduces the necessary sample size and helps to overcome issue (ii) by concentrating solely on extracting task-relevant MI. Specifically, we propose a three-step process tailored to RL: (1) We introduce ***Skill-aware Mutual Information (SaMI)***, a smaller ground-truth MI that discriminates context embeddings according to skills by maximising the MI between context embedding, skills and trajectories. Additionally, we provide a theoretical proof of why introducing skills can make the ground-truth MI smaller, thus making it easier to optimise; (2) We propose a more data-efficient K-sample estimator, ***Skill-aware Noise Contrastive Estimation (SaNCE)***, used to optimise SaMI and reduce the negative sample space based on skills to overcome the log-K curse; (3) We provide a framework to show how a simple objective, SaMI, can enable Meta-RL agents to autonomously discover skills, and propose a practical skill definition and skill-aware trajectory sampling method for SaNCE.

We demonstrate empirically in MuJoCo [Todorov et al., 2012] and Panda-gym [Gallouédec et al., 2021] that SaMI improves the zero-shot generalisation performance in sets of previously unseen tasks with moderate and extreme difficulty. In the MuJoCo and Panda-gym benchmark, SaMI enhances two Meta-RL algorithms [Yu et al., 2020, Fu et al., 2021] by achieving higher returns/success rates, especially in moderate and extreme testing tasks. This indicates SaMI's advantage in encoding information that enables an agent to execute effective skills in downstream control tasks. SaNCE-based RL algorithms utilise smaller sample spaces while achieving improved downstream control performance, indicating their potential to overcome the $\log$-K curse. Visualisation of the learned context embeddings shows distinct clusters corresponding to different skills, indicating that SaMI is a step towards more robust and versatile RL agents.

2 Related works

Meta-RL. Meta-RL methods train an agent conditioned on context embeddings to improve generalisation to unseen tasks. As the key component of Meta-RL, the quality of the context embedding can significantly affect the agent's performance. Existing algorithms can be categorised into three types based on different context embeddings. In the first category, the context embedding is learned by minimising the downstream RL loss [Rakelly et al., 2019, Yu et al., 2020]. PEARL [Rakelly et al., 2019] learns probabilistic context embeddings by recovering the value function. Multi-task SAC + TE (TESAC) [Yu et al., 2020] uses the Task Embedding (TE) [Hausman et al., 2018] as input to

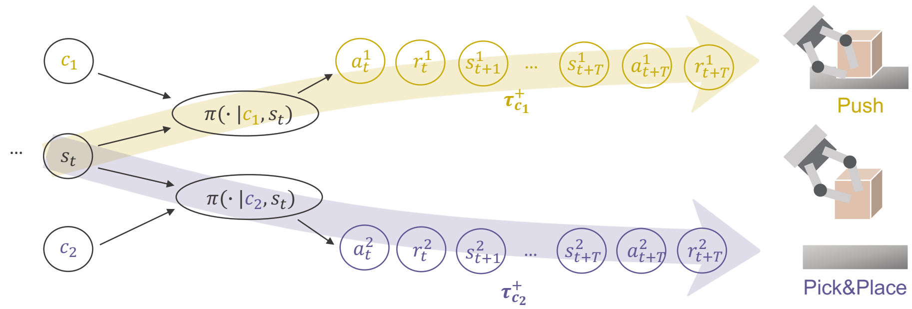


Figure 2: A policy π conditioned on a fixed context embedding c is defined as a skill $\pi(\cdot|c)$ (shortened as π_c). The policy π conditioned on a fixed c alters the state of the environment in a consistent way, thereby exhibiting a mode of skill. The skill $\pi(\cdot|c_1)$ moves the cube on the table in trajectory $\tau_{c_1}^+$ and is referred to as the Push skill; correspondingly, the Pick&Place skill $\pi(\cdot|c_2)$ takes the cube off the table and places it in the goal position in the trajectory $\tau_{c_2}^+$.

policies, parameterising the learned policies via a shared embedding space and maximising average returns. However, the update signals from the RL loss are stochastic and weak, and may not capture the similarity relations among tasks [Fu et al., 2021]. The second category involves learning context embeddings through dynamics prediction [Lee et al., 2020, Zhou et al., 2019], which can make the context embeddings noisy, as they may model irrelevant dependencies and overlook task-specific information [Fu et al., 2021]. The third category employs contrastive learning [Fu et al., 2021, Wang et al., 2021, Li et al., 2021, Sang et al., 2022], achieving significant improvements in context learning. However, these methods overlook the similarity of skills between different tasks, thus failing to achieve effective zero-shot generalisation by executing different skills. Our improvements build upon this third category by distinguishing context embeddings according to different optimal skills.

Contrastive learning. Contrastive learning has been applied to RL due to its significant momentum in representation learning in recent years, attributed to its superior effectiveness [Tishby and Zaslavsky, 2015, Hjelm et al., 2019, Dunion and Albrecht, 2024], ease of implementation [Oord et al., 2019], and strong theoretical connection to mutual information (MI) estimation [Poole et al., 2019]. MI is often estimated through noise-contrastive estimation (NCE) [Gutmann and Hyvärinen, 2010], InfoNCE [Oord et al., 2019], and variational objectives [Hjelm et al., 2019]. InfoNCE has garnered recent interest due to its lower variance [Song and Ermon, 2020] and superior performance in downstream tasks. However, InfoNCE may underestimate true MI because it is constrained by the number of samples K. To address this issue, CCM [Fu et al., 2021] leverages InfoNCE to learn the context embedding with a large number of samples. DOMINO [Mu et al., 2022] reduces the total MI by introducing an independence assumption; however, this results in biased information. Correspondingly, we focus on proposing an unbiased alternative MI objective and a more data-efficient K-sample estimator tailored for downstream RL tasks, which, to our knowledge, have not been addressed in previous research.

3 Preliminaries

Reinforcement learning. In Meta-RL, we assume an *environment* is a distribution $\xi(e)$ of *tasks* e (e.g. uniform in our experiments). Each task $e \sim \xi(e)$ has a similar structure that corresponds to a Markov Decision Process (MDP) [Puterman, 2014], defined by $\mathcal{M}_e = (\mathcal{S}, \mathcal{A}, R, P_e, \gamma)$, with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a reward function $R(s_t, a_t)$ where $s_t \in \mathcal{S}$ and $a_t \in \mathcal{A}$, state transition dynamics $P_e(s_{t+1}|s_t, a_t)$, and a discount factor $\gamma \in [0, 1)$. In order to address the problem of zero-shot generalisation, we consider the transition dynamics $P_e(s_{t+1}|s_t, a_t)$ vary across tasks $e \sim \xi(e)$ according to multiple *environmental features* $e = \{e^0, e^1, ..., e^N\}$ that are not included in states s and can be continuous random variables, such as mass and friction, or discrete random variables, such as the cube's material. For instance, in a cube-moving environment (Figure 1), an agent has different tasks that are defined by different environmental features (e.g., mass and friction). The Meta-RL agent's goal is to learn a generalisable policy π that is robust to such dynamic changes. Specifically, given a set of training tasks e sampled from $\xi_{\text{train}}(e)$, we aim to learn a policy that can maximise the discounted returns, $\arg\max_\pi \mathbb{E}_{e\sim\xi_{\text{train}}(e)}[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|a_t \sim \pi(a_t|s_t), s_{t+1}\sim P_e(s_{t+1}|s_t,a_t)]$, and can produce accurate control for unseen test tasks e sampled from $\xi_{\text{test}}(e)$.

Contrastive learning. In a Meta-RL setting, the context encoder $\psi(c|\tau_{0:t})$ first takes the trajectory $\tau_{c,0:t} = \{s_0, a_0, r_0, ..., s_t\}$ from the current episode as input and compresses it into a context embedding c. Then, the policy π, conditioned on context embedding c, consumes the current state s_t, outputs the action a_t. The policy π conditioned on a fixed c alters the state of the environment in a consistent way, thereby exhibiting a mode of skill. As a key component, the embedding c generated by the context encoder ψ for a task directly determines how the policy behaves. MI is a good measure of compression [Goldfeld et al., 2019], hence we focus on a context encoder that optimises the InfoNCE objective $I_{\text{InfoNCE}}(x;y)$, which is a K-sample estimator and lower bound of the MI $I(x;y)$ [Oord et al., 2019]. Given a query x and a set $Y = \{y_1, ..., y_K\}$ of K random samples containing one positive sample y_1 and $K-1$ negative samples from the distribution $p(y)$, $I_{\text{InfoNCE}}(x;y)$ is obtained by comparing pairs sampled from the joint distribution $x, y_1 \sim p(x,y)$ to pairs x, y_k built using a set of negative examples $y_{2:K}$:

$$I_{\text{InfoNCE}}(x;y|\psi,K) = \mathbb{E}\left[\log \frac{f_\psi(x,y_1)}{\frac{1}{K}\sum_{k=1}^{K} f_\psi(x,y_k)}\right]. \tag{1}$$

InfoNCE constructs a formal lower bound to the MI, i.e., $I_{\text{InfoNCE}}(x;y|\psi,K) \leq I(x;y)$ [Guo et al., 2022, Chen et al., 2021]. Given two inputs x and y, their *embedding similarity* is $f_\psi(x,y) = e^{\psi(x)^\top\cdot\psi(y_1)/\beta}$, where ψ is the context encoder that projects x and y into the context embedding space, the dot product is used to calculate the similarity score between $\psi(x), \psi(y)$ pairs [Wu et al., 2018, He et al., 2020], and β is a temperature hyperparameter that controls the sensitivity of the product. Some previous Meta-RL methods [Lee et al., 2020, Mu et al., 2022] learn a context embedding c by maximising $I_{\text{InfoNCE}}(c;\tau_c|\psi,K)$ between the context c embedded from a trajectory in the current task, and the historical trajectories τ_c under the same environmental features setting.

4 Skill-aware mutual information optimisation for Meta-RL

4.1 The log-K curse of K-sample MI estimators

In this section, we provide a theoretical analysis of the challenge inherent in learning a K-sample estimator for MI, commonly referred to as the *log-K* curse. Based on this theoretical analysis, we give insights to overcome this challenge. Given that we focus on the generalisation of RL, we only consider cases with a finite sample size of K. If a context encoder ψ in Equation 1 has sufficient training epochs, then $I_{\text{InfoNCE}}(x;y|\psi,K) \approx \log K$ [Mnih and Teh, 2012, Guo et al., 2022]. Hence, the MI we can optimise is bottlenecked by the number of available samples, formally expressed as:

Lemma 1 *Learning a context encoder ψ with a K-sample estimator and finite sample size K, we always have $I_{InfoNCE}(x;y|\psi,K) \leq \log K \leq I(x;y)$, when $x \not\perp\!\!\!\perp y$. (see proof in Appendix A)*

We do not consider the case when $x \perp\!\!\!\perp y$, i.e., $\log K \geq I(x;y) = 0$ $(\forall K \geq 1)$, because a meta-RL agent learns a context encoder by maximising MI between trajectories τ_c and context embeddings c, which are not independent according to the MDP graph in Figure 2. Good compression is crucial for generalisation, and compressing valuable information from a limited number of K samples requires MI as an effective measure of compression [Goldfeld et al., 2019]. We derive three key insights when learning a context encoder with finite sample size: (1) focus on a ground-truth MI that is smaller than $I(x;y)$; (2) develop a K-sample estimator tighter than the I_{InfoNCE}; (3) increasing sample quantity K, however, this is usually impractical. A meta-RL agent learns a context encoder by maximising MI between trajectories τ_c and context embeddings c. Driven

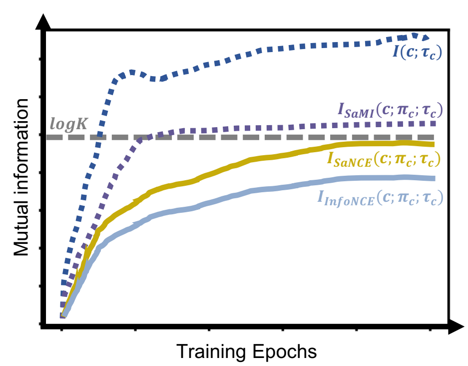


Figure 3: $I_{\text{InfoNCE}(c;\pi_c;\tau_c)}$, with a finite sample size of K, is a loose lower bound of $I(c;\tau_c)$ and leads to lower performance embeddings. $I_{\text{SaMI}}(c;\pi_c;\tau_c)$ is a lower ground-truth MI, and $I_{\text{SaNCE}}(c;\pi_c;\tau_c)$ is a tighter lower bound.

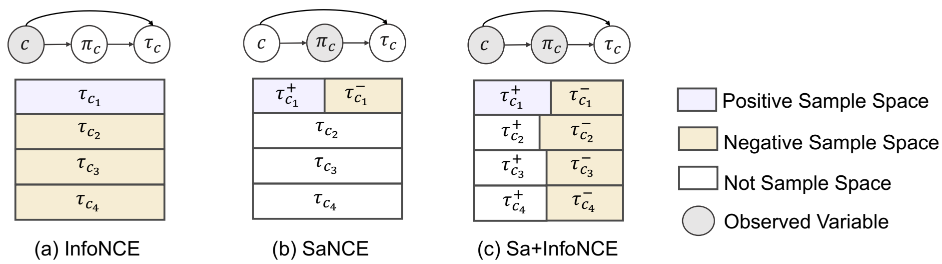


Figure 4: A comparison of sample spaces for task e_1. Positive samples τ_{c_1} or $\tau^+_{c_1}$ are always from current task e_1. For SaNCE, in a task e_k with embedding c_k, the positive skill $\pi^+_{c_k}$ conditions on c_k and generates positive trajectories $\tau^+_{c_k}$, and the negative skill $\pi^-_{c_k}$ generates negative trajectories $\tau^-_{c_k}$. The top graphs show the relationship between c, π_c and τ_c.

by insight (1), we introduce Skill-aware Mutual Information (SaMI) in Section 4.2, designed to enhance the zero-shot generalisation of downstream RL tasks. Corresponding to insight (2), we propose Skill-aware Noise Contrastive Estimation (SaNCE) to maximise SaMI with finite samples in Section 4.3. Finally, Section 4.4 demonstrates how to equip a Meta-RL agent with SaNCE in practice.

4.2 Skill-aware mutual information: a smaller ground-truth MI

A useful tool in learning a versatile agent is to understand when to explore novel skills or switch between existing skills in multi-task settings. To start with, we define skills [Eysenbach et al., 2018]:

Definition 1 (Skills) *A policy π conditioned on a fixed context embedding c is defined as a skill $\pi(\cdot|c)$, abbreviated as π_c. If a skill π_c is conditioned on a state s_t, we can sample actions $a_t \sim \pi(\cdot|c, s_t)$. After sampling actions from π_c at consecutive timesteps, we obtain a trajectory $\tau_{c,t:t_T} = \{s_t, a_t, r_t, s_{t+1}, \ldots, s_{t+T}, a_{t+T}, r_{t+T}\}$ which demonstrates a consistent mode of behaviour.*

After interacting with the environment, an agent should infer the task (i.e., environmental features $e = e^0, e^1, \ldots, e^N$) and adapt within an episode. The context encoder ψ should learn by maximising the MI between context embedding c, skills π_c, and trajectories τ_c. We achieve this learning process by maximising the MI $I_{\text{SaMI}}(c; \pi_c; \tau_c)$, in which we introduce a variable, skill π_c, into $I(c; \tau_c)$. Formally, we propose a novel MI optimisation objective for a context encoder, **Skill-aware Mutual Information (SaMI)**, which is defined as:

$$I_{\text{SaMI}}(c; \pi_c; \tau_c) = \mathbb{E}_{p(c,\pi_c,\tau_c)} \left\{ \log \frac{p(c, \pi_c, \tau_c)}{p(c)p(\pi_c)p(\tau_c)} \right\}. \tag{2}$$

Although we cannot evaluate $p(c, \pi_c, \tau_c)$ directly, we approximate it by Monte-Carlo sampling, using K samples from $p(c, \pi_c, \tau_c)$. A context encoder ψ trained with the objective of maximising MI $I_{\text{SaMI}}(c; \pi_c; \tau_c)$ will converge more quickly because $I_{\text{SaMI}}(c; \pi_c; \tau_c) \leq I(c; \tau_c)$ (see proof in Appendix B). I_{SaMI} is an objective that can discriminate between skills, therefore it also enables RL agents to autonomously discover diverse skills.

4.3 Skill-aware noise contrastive estimation: a tighter K-sample estimator

Despite InfoNCE's success as a K-sample estimator for approximating MI [Laskin et al., 2020, Eysenbach et al., 2022], its learning efficiency plunges due to limited numerical precision, which is called the $\log$-K curse, i.e., $I_{\text{InfoNCE}} \leq \log K \leq I_{\text{SaMI}}$ [Chen et al., 2021] (see proof in Appendix B). When $K \to +\infty$, we can expect $I_{\text{InfoNCE}} \approx \log K \approx I_{\text{SaMI}}$ [Guo et al., 2022]. However, increasing K is too expensive, especially in complex environments with enormous negative sample space. In response, we propose a novel K-sample estimator with a reduced required sample space size K^* ($K^* \ll +\infty$). First, we define K^*:

Definition 2 (K*) *$K^* = |c| \cdot |\pi_c| \cdot M$ is defined as the number of trajectories in the replay buffer (i.e., the sample space), in which $|c|$ represents the number of different context embeddings c, $|\pi_c|$ represents the number of different skills π_c, and M is a natural number.*

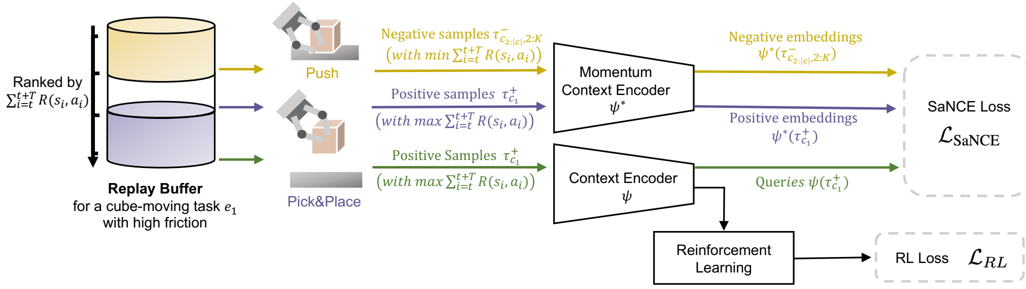


Figure 5: A practical framework for using SaNCE in the meta-training phase. During meta-training, we sample trajectories from the task-specific replay buffer for off-policy training. Queries are generated by a context encoder ψ, which is updated with gradients from both the SaNCE loss $\mathcal{L}_{\text{SaNCE}}$ and the RL loss $\mathcal{L}_{RL}$. negative/positive embeddings are encoded by a momentum context encoder ψ^*, which is driven by a momentum update with the encoder ψ. During meta-testing, the meta-trained context encoder ψ embeds the current trajectory, and the RL policy takes the embedding as input together with the state for adaptation within an episode.

Note that K^* is the maximum batch size that can be sampled in contrastive learning. Therefore, to ensure that I_{InfoNCE} is a tight bound of I_{SaMI}, we require that $I_{\text{InfoNCE}} \approx \log K \approx I_{\text{SaMI}}$ when $K \to K^*$. Under the definition of K^*, the replay buffer can be divided according to the different context embeddings c and skills π_c (i.e., observing context embeddings c and skills π_c). In real-world robotic control tasks, the sample space size significantly increases due to multiple environmental features $e = \{e^0, e^1, ..., e^N\}$. Taking the sample space of InfoNCE as an example (Figure 4(a)), in the current task e_1 with context embedding c_1, positive samples are trajectories τ_{c_1} generated after executing the skill π_{c_1} in task e_1, and negative samples are trajectories $\{\tau_{c_2}, ...\}$ from other tasks $\{e_2, ...\}$. The permutations and combinations of N environmental features lead to an exponential growth in task number $|c|$, which in turn results in an increase of sample space $K^*_{\text{InfoNCE}} = |c| \cdot |\pi| \cdot M$.

We introduce a tight K-sample estimator, **Skill-aware Noise Contrastive Estimation (SaNCE)**, which is used to approximate $I_{\text{SaMI}}(c; \pi_c; \tau_c)$ with a reduced K^*_{SaNCE}. For SaNCE, both positive samples $\tau^+_{c_1}$ and negative samples $\tau^-_{c_1}$ are sampled from the current tasks e_1, but are generated by executing positive skills $\pi^+_{c_1}$ and negative skills $\pi^-_{c_1}$, respectively. Here, a *positive skill* is intuitively defined by whether it is optimal for the current task e, with a more formal definition provided in Section 4.4. For instance, in a cube-moving task under a large friction setting, the agent executes a skill π^+_c after several iterations of learning, and obtains corresponding trajectories τ^+_c where the cubes leave the table surface. This indicates that the skill π^+_c is Pick&Place and other skills π^-_c may include Push or Flip (flipping the cube to the goal position), with corresponding trajectories τ^-_c where the cube remains stationary or rolls on the table. Formally, we can optimise the K-sample lower bound I_{SaNCE} to approximate I_{SaMI}:

$$
\begin{aligned}
& I_{\text{SaNCE}}(c; \pi_c; \tau_c | \psi, K) \\
&= \mathbb{E}_{p(c_1, \pi_{c_1}, \tau^+_{c_1}) p(\tau^-_{c_1, 2:K})} \left[\log \left(\frac{K \cdot f_\psi(c_1, \pi_{c_1}, \tau^+_{c_1})}{f_\psi(c_1, \pi_{c_1}, \tau^+_{c_1}) + \sum_{k=2}^{K} f_\psi(c_1, \pi_{c_1}, \tau^-_{c_1, k})} \right) \right] \\
&\leq I_{\text{SaMI}}(c; \pi_c; \tau_c)
\end{aligned} \tag{3}
$$

where $f_\psi(c_1, \pi_{c_1}, \tau_{c_1}) = e^{\psi(\tau_{c_1})^\top \cdot \psi^*(\tau_{c_1})/\beta}$. The query $c_1 = \psi(\tau_{c_1})$ is generated by the context encoder ψ. For training stability, we use a momentum encoder ψ^* to produce the positive and negative embeddings. SaNCE significantly reduces the required sample space size K^*_{SaNCE} by sampling trajectories τ_c based on different skills π_c (Figure 4(b)) in task e_1, so that $K^*_{\text{SaNCE}} = |c| \cdot |\pi_c| \cdot M = |\pi_{c_1}| \cdot M \leq K^*_{\text{InfoNCE}}$ ($|c| = |c_1| = 1$). Therefore, I_{SaNCE} satisfies Lemma 2:

Lemma 2 *With a context encoder ψ and finite sample size K, we have $I_{InfoNCE}(c; \pi_c; \tau_c | \psi, K) \leq I_{SaNCE}(c; \pi_c; \tau_c | \psi, K) \leq \log K \leq I_{SaMI}(c; \pi_c; \tau_c) \leq I(c; \tau_c)$. (see proof in Appendix B)*

SaNCE is a plug-and-play module for any other NCE. For example, the negative sample space can be more diverse by combining SaNCE and InfoNCE with $K^*_{\text{Sa+InfoNCE}} = \left(\sum_{i=1}^{|c|} |\pi^-_{c_i}| + |\pi^+_{c_1}|\right) \cdot M$ (Figure 4(c)). A detailed analysis of the sample size of $I_{\text{Sa+InfoNCE}}$ can be found in Appendix C.

4.4 Skill-aware trajectory sampling strategy

Methods that focus on skill diversity often rely heavily on accurately defining and identifying skills [Eysenbach et al., 2018], with some requiring a prior skill distribution that is often inaccessible [Shi et al., 2022]. This variability in skill definitions across tasks can affect the robustness and generalisability of these methods. For these reasons, our approach does not use such skill definitions and priors for specific environments or tasks. In this section, we give the distinctiveness of skills and propose a practical trajectory sampling method.

In this study, we believe that distinctiveness of skills is inherently difficult to achieve — a slight difference in states can make two skills distinguishable, and not necessarily in a semantically meaningful way. Instead, we should focus on whether skills acquired by an agent can complete a task. For example, in high-friction tasks, the agent must acquire the Pick&Place skill to avoid large frictional forces, while in high-mass tasks, the agent must learn the Push skill since it cannot lift the cube. In that way, without skills definition in a semantically meaningful way, we only need to train an agent on a set of tasks, and it will autonomously discover diverse skills to work across multiple tasks.

We consider the task-specific definition of positive/negative samples. In a given task e, *positive skills* π^+_c are defined as the optimal skills achieving the highest return $\sum_{i=t}^{t+T} R(s_i, a_i)$, while *negative skills* π^-_c are those with lower returns. Thus, we can simply sample the trajectory with the ranked highest return as the positive sample τ^+_c, and the one with the lowest return as the negative sample τ^-_c. Positive samples are generated by the optimal skill for the current task, while lower return samples are classified as negative. This polarised definition helps the model select the optimal skill from among many skills with varying returns and avoids the issue of hard negative examples during training Robinson et al. [2021]. The SaNCE loss is then minimised to bring the context embeddings of the highest return trajectories closer in the embedding space while distancing those of negative trajectories. Note that, at the end of the training, the top-ranked trajectories in the ranked replay buffer correspond to positive samples τ^+_c with high returns, and the lower-ranked ones are negative samples τ^-_c with low returns. However, before the agent is able to achieve high returns, all trajectories are with low returns. Therefore, our SaNCE loss is a soft version of the K-sample SaNCE estimator:

$$\mathcal{L}_{\text{SaNCE}} = -\max\left(||\psi(\tau^+_c), \psi(\tau^-_c)||_{L2}, 1\right) \cdot I_{\text{SaNCE}} \tag{4}$$

where $||\cdot||_{L2}$ represents the Euclidean distance [Tabak, 2014]. Figure 5 provides a practical framework of SaNCE, with a cube-moving example task e_1 under high friction. In task e_1, the positive skill $\pi^+_{c_1}$ is the Pick&Place skill, which is used to generate queries $\psi(\tau^+_{c_1})$ and positive embeddings $\psi^*(\tau^+_{c_1})$; after executing Push skill we get negative samples $\tau^-_{c_1}$ and negative embeddings $\psi^*(\tau^-_{c_1})$.

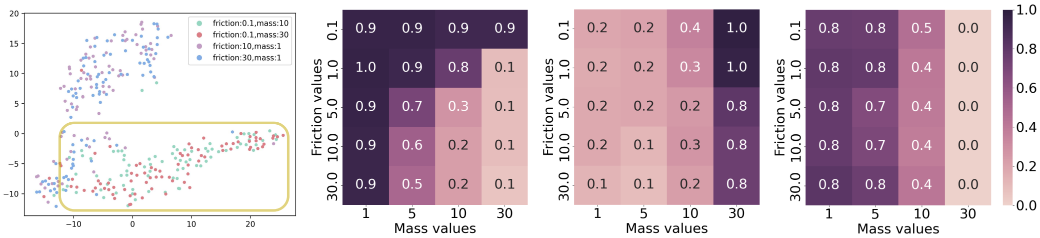
5 Experiments

We use a three-step process to demonstrate the benefits of SaMI in each environment and answer three questions: (1) Does optimising SaMI lead to increased returns during training and zero-shot generalisation (see Table 1 and 2)?; (2) Does SaMI help the RL agents to be versatile and embody multiple skills (see Figure 6)?; (3) Can SaNCE overcome the log-K curse in sample-limited scenarios (see Table 1 and 2, and Section 5.4)?

5.1 Experimental setup

Modified benchmarks with multiple environmental features. [1] We demonstrate the efficacy of our method using two benchmarks, Panda-gym [Gallouédec et al., 2021] and MuJoCo [Todorov et al., 2012] (details in Sections 5.2 and 5.3). The benchmarks are modified to be influenced by multiple environmental features simultaneously. Environmental features are sampled at the start of each episode during both the meta-training and meta-testing phases. During meta-training, we

[1] Our modified benchmarks are open-sourced at Anonymous Link



(a) t-SNE visualisation (b) Success rate (c) Push probability (d) Pick&Place probability

Figure 6: (a) t-SNE visualisation of context embeddings in the Panda-gym environment. The clustered points in the yellow box correspond to the Push skill executed in scenarios with a large mass. Heatmap of (b) success rate, and probability of learned (c) Push skill and (d) Pick&Place skill of SaCCM. Under large mass scenarios, the probability of executing Push skill is higher than Pick&Place skill.

uniform-randomly select a combination of environmental features from a training task set. At test time, we evaluate each algorithm in unseen tasks with different environmental features outside the training range. Generalisation performance is measured in two different regimes: moderate and extreme. The moderate regime draws environmental features from a closer range to the training range compared to the extreme. For all our experiments, we report the mean and standard deviation of the models trained over five seeds in both training and test tasks. Further experimental details are available in Appendix D.

Baselines. In our experiments, we primarily compare InfoNCE with SaNCE to demonstrate the performance improvements brought by SaNCE. Therefore, we compare our method with three prevailing and competitive baselines. First, we consider CCM [Fu et al., 2021], which is equipped with InfoNCE. Additionally, we consider TESAC [Yu et al., 2020], which employs a value function loss, allowing us to evaluate the impact on the context encoder without contrastive loss. Given that CCM and TESAC are using RNN encoder, we also consider PEARL [Rakelly et al., 2019], which utilises an MLP context encoder and a similar loss as TESAC. In Appendix G, we also provide a comparison with DOMINO [Mu et al., 2022] and CaDM [Lee et al., 2020] using exactly the same environmental setting in MuJoCo benchmark.

Our methods. [2] We employ Soft Actor-Critic (SAC) [Haarnoja et al., 2018] as the base RL algorithm and trained agents for 1.6 million timesteps in each environment (please refer to Appendix D.3 for more implementation details). SaNCE is a simple objective based on mutual information that can be used to train any context encoder. Two RL algorithms are equipped with SaNCE: (1) SaTESAC is TESAC with SaNCE, which uses SaNCE for contrastive learning, with a $|c|$ times smaller sample space than that of other algorithms, as shown in Figure 4(b); (2) SaCCM is CCM with SaNCE, where the contrastive learning combines InfoNCE and SaNCE, as shown in Figure 4(c).

5.2 Panda-gym

Task description. Our modified Panda-gym benchmark contains a robot arm control task using the Franka Emika Panda [Gallouédec et al., 2021], where the robot needs to move a cube to a target position. Unlike previous works, we simultaneously modify multiple environmental features (cube mass and table friction) that characterise the transition dynamics, and the robot can flexibly execute different skills (Push and Pick&Place) for different tasks. This environment demands high skill diversity from the agent. For example, in high-friction tasks, the agent must use the Pick&Place skill, while in high-mass tasks, it must use the Push skill.

Table 1: Comparison of success rate with baselines in Panda-gym (over 5 seeds). **Bold text** signifies the highest average return. $*$ next to the number means that the algorithm with SaMI has statistically significant improvement over the same algorithm without SaMI. All significance claims based on t-tests with significance threshold of $p < 0.05$.

	Training	Test (moderate)	Test (extreme)
PEARL	0.42±0.19	0.10±0.06	0.11±0.05
TESAC	0.50±0.22	0.31±0.20	0.22±0.21
CCM	0.80±0.19	0.49±0.23	0.29±0.28
SaTESAC	**0.92±0.04**$*$	**0.56±0.24**$*$	**0.37±0.34**$*$
SaCCM	**0.93±0.05**$*$	**0.57±0.26**$*$	**0.36±0.35**$*$

[2] Our code, video demos and experimental data are available at https://anonymous.4open.science/r/SaMI

Table 2: Comparison of average return ± standard deviation with baselines in modified MuJoCo benchmark (over 5 seeds). **Bold number** signifies the highest return. $*$ next to the number means that the algorithm with SaMI has statistically significant improvement over the same algorithm without SaMI. All significance claims based on t-tests with significance threshold of $p < 0.05$.

	Crippled Ant			Crippled Half-cheetah		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
PEARL	1682±73	996±21	888±31	1998±973	698±548	746±1092
TESAC	2139±90	1952±40	1048±124	3967±955	874±901	846±849
CCM	2361±114	2047±83	1527±301	3481±488	821±575	873±914
SaTESAC	2638±406	2379±528	**2131±132***	4328±1092	**1143±664***	**1540±1094**
SaCCM	2355±170	2310±314	2007±68*	**4478±1131***	1007±568	1027±782
	Ant			Half-cheetah		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
PEARL	5153±581	3873±235	3802±409	5802±773	2190±970	1346±692
TESAC	6789±451	4705±279	4108±369	6298±2310	3173±1210	1159±338
CCM	6901±567	5179±902	4700±696	6955±788	3963±622	1325±269
SaTESAC	7314±545	5513±648*	4940±531*	7430±1026	4058±890	1780±102*
SaCCM	7478±539	5717±488	5215±377	7154±965	3849±689	1926±218*
	SlimHumanoid			HumanoidStandup		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
PEARL	6947±3541	3697±2674	2018±907	95456±13445	63242±13546	64224±15467
TESAC	8437±1798	6989±1301	3760±308	158384±14455	153944±15046	74220±19980
CCM	7696±1907	5784±531	2887±1058	146480±33745	154601±16291	94991±15258
SaTESAC	**10216±1620**	**7886±2203**	6123±1403*	178142±10081*	168337±12123	133335±24607*
SaCCM	9312±705	7430±1587	**6473±2001***	**187930±19338***	**181033±14628**	**141750±27426**
	Hopper			Crippled Hopper		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
PEARL	934±242	874±366	799±298	3091±298	2387±656	456±235
TESAC	1492±59	1499±35	1459±72	3575±192	3298±551	722±161
CCM	1484±54	1446±64	1452±58	3455±301	3409±239	1009±289
SaTESAC	1502±20	1453±39	1447±14	3391±84	3262±166	1839±130
SaCCM	1462±45	1462±14	1451±67	3449±103	3390±211	**2059±221***
	Walker			Crippled Walker		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
PEARL	7524±2455	3355±2555	1984±356	7899±2532	4377±2563	2965±1426
TESAC	7747±1772	4355±1530	2581±407	9908±1561	5929±1971	3041±912
CCM	8136±557	5476±803	2519±682	10317±1137	6233±1869	3098±821
SaTESAC	8675±752	5840±676	**3632±404***	10389±1031	**8387±1291***	4280±485*
SaCCM	8361±586	5779±691	3481±332*	10496±951	8235±1212	**4824±839***

Results and skill analysis. As shown in Table 1, SaTESAC and SaCCM achieve superior generalisation performance compared to PEARL, TESAC, and CCM, with a smaller sample space. Faced with an unknown task, the agent achieved the three steps of "explore effectively, infer, adapt," and acquired multiple skills (Push, Pick&Place, Drag and Slide) to work across various tasks (please refer to the video demos[2]). We used t-SNE [Van der Maaten and Hinton, 2008] and PCA [Jolliffe, 2002] to visualise the context embedding. We plotted the context embedding at the final time step in 100 tests for each task, which can compress how a skill alters the state of the environment in a consistent way. Additionally, we determined the skills by detecting contact points between the end effector and the cube, and between the cube and the table (see Appendix D for more details), and then employed heatmaps [Waskom, 2021] to visualise executed skills. As shown in Figure 6, when the cube mass is large (30 Kg and 10 Kg), the agent learned the Push skill (corresponding to the clustered points in the yellow bounding box in Figure 6(a)). With smaller masses, the agent learned the Pick&Place skill. However, as shown in Figure 15 in Appendix F.1, CCM did not exhibit clear skill grouping. Based on the t-test results in Table 1, SaMI significantly improved the success rate in the training, moderate test, and extreme test sets at a significance level of 0.05. Overall, SaMI helps to compress high-quality skill-related information from the trajectories and helps to acquire diverse skills autonomously. More visualisation results can be found in Appendix F.

5.3 MuJoCo

Task description. We extended the modified MuJoCo benchmark introduced in DOMINO [Mu et al., 2022] and CaDM [Lee et al., 2020]. It contains ten typical robotic control environments based on the MuJoCo physics engine [Todorov et al., 2012]. Hopper, Walker, Half-cheetah, Ant, HumanoidStandup, and SlimHumanoid are influenced by continuous environmental features (i.e., mass, damping) that affect transition dynamics. Crippled Ant, Crippled Hopper, Crippled Walker, and Crippled Half-cheetah are more challenging due to the addition of discrete environmental features (i.e., randomly crippled leg joints), requiring agents to master different skills (e.g., switching from running to crawling after a leg is crippled).

Results and skill analysis. Table 2 shows the average return of our method and baselines on both training and test tasks. SaTESAC and SaCCM gained higher returns in both training and testing in most of the tasks, in which the Ant, Half-Cheetah and Hopper tasks are exceptions. Ant and Hopper robots only need to learn a single skill to generalise across different tasks. For example, the Hopper robot learned to hop forward on the floor to adapt to different mass values. When environments (Crippled Ant, Crippled Hopper, Crippled Half-Cheetah, SlimHumanoid, HumanoidStandup, and Crippled Walker) become complex and require diverse skills for different tasks, SaNCE brings significant improvements. For instance, when the Ant robot has 3 or 4 legs available, it learns to roll to work across varying mass and damping. However, during zero-shot generalisation, when only 2 legs are available, the ant robot can no longer roll and it adapts by walking using its 2 healthy legs. This aligns with the result of the t-test [Rice and Rice, 2007] in Table 2, which SaMI brings significant improvement on the extreme test set (more details refer to Appendix H). Therefore, i) SaMI helps the RL agents to be versatile and embody multiple skills; ii) SaMI leads to increased returns during training and zero-shot generalisation, especially in environments that require different skills. Please refer to our video demos [2] for different skills in all environments, and visualisation results in Appendix F.2.

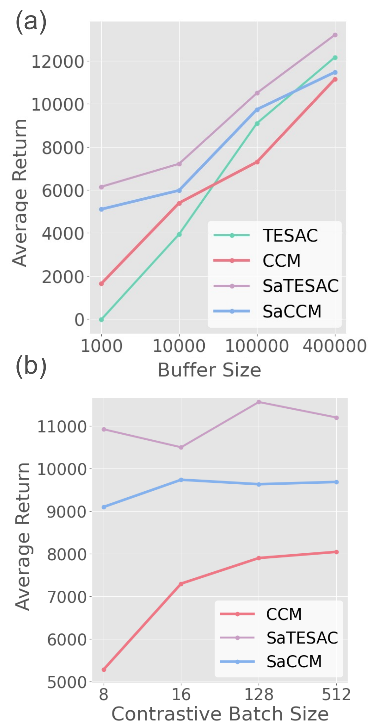


Figure 7: Changes of average return in the SlimHumanoid environment. (a) The impact of buffer size on RL control performance (i.e., TESAC, CCM, SaTESAC, SaCCM). (b) The impact of contrastive batch size on the contrastive context encoder (i.e., CCM, SaTESAC, SaCCM).

5.4 Analysis of the $\log$-K curse in sample-limited scenarios

In this section, we further analyse whether SaNCE can overcome the $\log$-K curse. During the training phases, we sample the environmental features at the beginning of each episode. Therefore, throughout the training process, the context encoder needs to learn the context embedding distribution of multiple tasks. Because InfoNCE requires sampling negative samples across all tasks, and SaNCE samples negative samples from the current task, SaNCE's negative sample space is $|c|$ times smaller than that of InfoNCE. For example, where both mass and damping have five values, SlimHumanoid environment has a maximum of 25 different tasks, making the sampling space of InfoNCE potentially 25 times larger than that of SaNCE. According to Table 1 and 2, RL algorithms equipped with SaNCE can achieve better or comparable performance with a much smaller number of negative samples K than InfoNCE. This indicates that SaNCE indeed plays a significant role in addressing the $\log$-K curse, and the SaMI objective helps the contrastive context encoder extract information that is crucial for downstream RL tasks.

Furthermore, the number of negative samples K also relates to two hyperparameters: ***buffer size***, which directly determines the size of the negative sample space; and ***contrastive batch size***, which determines the number of samples used for training contrastive context encoder at each update step. Therefore, we conducted further analysis on these two hyperparameters. As shown in Figure 7, we find that reductions in buffer size and contrastive batch size do not significantly decrease the average return for SaCCM and SaTESAC, which exhibit state-of-the-art performance with small buffers and contrastive batch sizes. Note that the results in Table 2 correspond to a buffer size of 100,000 and a contrastive batch size of 12. Experimental results for all environments (Appendix E.2) further demonstrate that SaNCE is not highly sensitive to changes in K, and indeed shows great potential in overcoming the $\log$-K curse.

6 Conclusion and future work

In this paper, we propose a Skill-aware Mutual Information (SaMI) to learn context embeddings for zero-shot generalisation in downstream RL tasks, and a Skill-aware Noise Contrastive Estimation (SaNCE) to optimise SaMI and overcome the $\log$-K curse. RL algorithms equipped with SaMI have achieved state-of-the-art performance in the MuJoCo and Panda-gym benchmarks. Through skill analysis and video demos in Panda-gym and MuJoCo, we confirm that the context encoder, learned by maximising SaMI, can compress high-quality skill-related information from trajectories, thereby assisting the RL agent acquire diverse skills autonomously and zero-shot generalising across various tasks. Notably, the optimisation process of SaNCE utilises a far smaller negative sample space than baselines. Coupled with further experimental analysis of buffer size and contrastive batch size on MuJoCo, we demonstrate that SaNCE helps overcome the $\log$-K curse. Our results indicate the importance of aligning the optimisation objectives of representation learning and downstream optimal control to enhance task performance and improve data efficiency.

Given that environmental features are often interdependent, such as a cube's material correlating with friction and mass, SaMI does not introduce independence assumptions like DOMINO [Mu et al., 2022]. Therefore, future work will focus on verifying and enhancing SaMI's potential in more complex tasks where environmental features are correlated. This will contribute to our ultimate goal: developing a generalist and versatile agent capable of working across multiple tasks and even real-world tasks in the near future.

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*, 2021.

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019.

Junya Chen, Zhe Gan, Xuan Li, Qing Guo, Liqun Chen, Shuyang Gao, Tagyoung Chung, Yi Xu, Belinda Zeng, Wenlian Lu, et al. Simpler, faster, stronger: Breaking the log-k curse on contrastive learners with flatnce. *arXiv preprint arXiv:2107.01152*, 2021.

Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019a.

Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019b.

Remi Tachet des Combes, Philip Bachman, and Harm van Seijen. Learning invariances for policy generalization, 2018. URL `https://openreview.net/forum?id=BJHRaK1PG`.

Mhairi Dunion and Stefano V Albrecht. Multi-view disentanglement for reinforcement learning with multiple cameras. In *Reinforcement Learning Conference*, 2024.

Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V Albrecht. Conditional mutual information for disentangled representations in reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a.

Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V Albrecht. Temporal disentanglement of representations for improved generalisation in reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023b.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function, 2018. URL `https://arxiv.org/abs/1802.06070`.

Benjamin Eysenbach, Tianjun Zhang, Sergey Levine, and Russ R Salakhutdinov. Contrastive learning as goal-conditioned reinforcement learning. *Advances in Neural Information Processing Systems*, 35:35603–35620, 2022.

Jörg K.H. Franke, Gregor Koehler, André Biedenkapp, and Frank Hutter. Sample-efficient automated deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

Haotian Fu, Hongyao Tang, Jianye Hao, Chen Chen, Xidong Feng, Dong Li, and Wulong Liu. Towards effective context for meta-reinforcement learning: an approach based on contrastive learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7457–7465, 2021.

Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS*, 2021.

Samuel Garcin, James Doran, Shangmin Guo, Christopher G. Lucas, and Stefano V. Albrecht. DRED: Zero-shot transfer in reinforcement learning via data-regularised environment design. In *International Conference on Machine Learning (ICML)*, 2024.

Ziv Goldfeld, Ewout van den Berg, Kristjan Greenewald, Igor Melnyk, Nam Nguyen, Brian Kingsbury, and Yury Polyanskiy. Estimating information flow in deep neural networks, 2019. URL `https://arxiv.org/abs/1810.05728`.

Qing Guo, Junya Chen, Dong Wang, Yuewei Yang, Xinwei Deng, Jing Huang, Larry Carin, Fan Li, and Chenyang Tao. Tight mutual information estimation with contrastive fenchel-legendre optimization. *Advances in Neural Information Processing Systems*, 35:28319–28334, 2022.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

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*, pages 1861–1870. PMLR, 2018.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks, 2016.

Ian T Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International conference on machine learning*, pages 5639–5650. PMLR, 2020.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pages 5757–5766. PMLR, 2020.

Lanqing Li, Yuanhao Huang, Mingzhe Chen, Siteng Luo, Dijun Luo, and Junzhou Huang. Provably improved context-based offline meta-rl with attention and contrastive learning. *arXiv preprint arXiv:2102.10774*, 2021.

William McGill. Multivariate information transmission. *Transactions of the IRE Professional Group on Information Theory*, 4(4):93–111, 1954.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

Yao Mu, Yuzheng Zhuang, Fei Ni, Bin Wang, Jianyu Chen, Jianye HAO, and Ping Luo. DOMINO: Decomposed mutual information optimization for generalized context in meta-reinforcement learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

Kento Nozawa and Issei Sato. Understanding negative samples in instance discriminative self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34: 5784–5797, 2021.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2019.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR, 2019.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, pages 5331–5340. PMLR, 2019.

John A Rice and John A Rice. *Mathematical statistics and data analysis*, volume 371. Thomson/Brooks/Cole Belmont, CA, 2007.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples, 2021. URL `https://arxiv.org/abs/2010.04592`.

Tong Sang, Hongyao Tang, Yi Ma, Jianye Hao, Yan Zheng, Zhaopeng Meng, Boyan Li, and Zhen Wang. Pandr: Fast adaptation to new environments from offline experiences via decoupling policy and environment representations, 2022.

Younggyo Seo, Kimin Lee, Ignasi Clavera Gilaberte, Thanard Kurutach, Jinwoo Shin, and Pieter Abbeel. Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:12968–12979, 2020.

Lucy Xiaoyang Shi, Joseph J. Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In *Conference on Robot Learning*, 2022.

Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *International Conference on Learning Representations*, 2020.

John Tabak. *Geometry: the language of space and form*. Infobase Publishing, 2014.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Bernie Wang, Simon Xu, Kurt Keutzer, Yang Gao, and Bichen Wu. Improving context-based meta-reinforcement learning with self-supervised trajectory contrastive learning, 2021.

Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60): 3021, 2021. doi: 10.21105/joss.03021. URL `https://doi.org/10.21105/joss.03021`.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

Raymond W Yeung. A new outlook on shannon's information measures. *IEEE transactions on information theory*, 37(3):466–474, 1991.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies. In *International Conference on Learning Representations*, 2019.

A Proof of Lemma 1

Given a query x and a set $Y = \{y_1, ..., y_K\}$ of K random samples containing one positive sample y_1 and $K-1$ negative samples from the distribution $p(y)$, A K-sample InfoNCE estimator is obtained by comparing pairs sampled from the joint distribution $x, y_1 \sim p(x, y)$ to pairs x, y_k built using a set of negative examples $y_{2:K}$. InfoNCE is obtained by comparing positive pairs (x, y_1) and negative pairs (x, y_k), where $y_k \sim y_{2:K}$, as follows:

$$I_{\text{InfoNCE}}(x; y|\psi, K) = \mathbb{E}_{p(x,y_1)p(y_{2:K})}\left[\log\left(\frac{f_\psi(x, y_1)}{\frac{1}{K}\sum_{k=1}^{K} f_\psi(x, y_k)}\right)\right] \tag{5}$$

Step 1. Let us prove the K-sample InfoNCE estimator is upper bounded by $\log K$. According to Mu et al. [2022], $\frac{f_\psi(x,y_1)}{\sum_{k=1}^{K} f_\psi(x,y_k)} = \frac{f_\psi(x,y_1)}{f_\psi(x,y_1)+\sum_{k=2}^{K} f_\psi(x,y_k)} \leq 1$. So we have:

$$\begin{aligned}
I_{\text{InfoNCE}}(x; y|\psi, K) &= \mathbb{E}_{p(x,y_1)p(y_{2:K})}\left[\log\left(\frac{f_\psi(x, y_1)}{\frac{1}{K}\sum_{k=1}^{K} f_\psi(x, y_k)}\right)\right] \\
&= \mathbb{E}_{p(x,y)}\left[\mathbb{E}_{p(y_{2:K})}\log\left(\frac{K \cdot f_\psi(x, y_1)}{\sum_{k=1}^{K} f_\psi(x, y_k)}\right)\right] \\
&\leq \log K
\end{aligned} \tag{6}$$

Hence, we have $I_{\text{InfoNCE}}(x; y|\psi, K) \leq \log K$.

Step 2. We have the $I(x; y) \geq I_{\text{InfoNCE}}(x; y|\psi, K)$ which are rightmost and the leftmost side in Lemma 1 according to:

Proposition 1 *[Poole et al., 2019] A K-sample estimator is an asymptotically tight lower bound to the MI, i.e.,*

$$I(x; y) \geq I_{\textit{InfoNCE}}(x; y|\psi, K), \lim_{x\to+\infty} I_{\textit{InfoNCE}}(x; y|\psi, K) \to I(x; y)$$

Proof. See Poole et al. [2019] for a neat proof of how the multi-sample estimator (e.g., InfoNCE) lower bounds MI.

Step 3. In this research, the context encoder ψ in $f_\psi(x, y)$ is implemented using an RNN to approximate $\frac{p(y|x)}{p(y)}$ [Oord et al., 2019]. For most deep learning platforms, with a powerful learner for ψ and a finite size K such that $I(x; y) \geq \log K$, we can reasonably expect $I_{\text{InfoNCE}} \approx \log K$ after a few training epochs. Therefore, during training, when $K \ll +\infty$, we always have $I(x; y) \geq \log K$.

Proof. See Chen et al. [2021] for more detailed proof.

Step 4. Let us prove that the K-sample InfoNCE bound is asymptotically tight. The specific choice of context encoder ψ relates to the tightness of the K-sample NCE bound. InfoNCE [Oord et al., 2019] sets $f_\psi(x, y) \propto \frac{p(y|x)}{p(y)}$ to model a density ratio which preserves the MI between x and y, where $\propto$ stands for ‘proportional to’ (i.e. up to a multiplicative constant). Let us plug in

$f_\psi(x,y) = f^*_\psi(x,y) = \frac{p(y|x)}{p(y)}$ into InfoNCE, and we have

$$
\begin{aligned}
I_{\text{InfoNCE}}(x;y|\psi,K) &= \mathbb{E}\left[\log\left(\frac{f^*_\psi(x,y_1)}{\sum_{k=1}^K f^*_\psi(x,y_k)}\right)\right] + \log K \\
&= -\mathbb{E}\left[\log\left(1 + \frac{p(y)}{p(y|x)}\sum_{k=2}^{K}\frac{p(y_k|x)}{p(y_k)}\right)\right] + \log K \\
&\approx -\mathbb{E}\left[\log\left(1 + \frac{p(y)}{p(y|x)}(K-1)\mathbb{E}_{y_k\sim p(y)}\frac{p(y_k|x)}{p(y_k)}\right)\right] + \log K \\
&= -\mathbb{E}\left[\log\left(1 + \frac{p(y_1)}{p(y_1|x)}(K-1)\right)\right] + \log K \\
&\approx -\mathbb{E}\left[\log\frac{p(y)}{p(y|x)}\right] - \log(K-1) + \log K \\
&= I(x;y) - \log(K-1) + \log K
\end{aligned} \tag{7}
$$

Now taking $K \to +\infty$, the last two terms cancel out.

Putting it together. Combining $I(x;y) \geq \log K$ with Proposition 1 and Equation 6, we have Lemma 1:

$$
I_{\text{InfoNCE}}(x;y|\psi,K) \leq \log K \leq I(x;y). \tag{8}
$$

Besides, according to Equation 7, with sample size $K \to +\infty$, K-sample InfoNCE bound is sharp and approaches to the true MI $I(x;y)$, i.e., $I_{\text{InfoNCE}}(x;y|\psi,K) \approx \log K \approx I(x;y)$.

B Proof for Lemma 2

Step 1. According to Lemma 1, we have $I_{\text{InfoNCE}}(c;\pi_c;\tau_c|\psi,K) \leq \log K \leq I_{\text{SaMI}}(c;\pi_c;\tau_c)$ (shown in Figure 3).

Step 2. Let us prove SaNCE is a K-sample SaNCE estimator and upper bounded by $\log K$. Because $\frac{f_\psi(c,\pi_c,\tau_c^+)}{f_\psi(c,\pi_c,\tau_c^+)+\sum_{k=2}^K f_\psi(c,\pi_c,\tau_{c,k}^-)} \leq 1$ [Mu et al., 2022], so we have:

$$
\begin{aligned}
&I_{\text{SaNCE}}(c;\pi_c;\tau_c|\psi,K) \\
&= \mathbb{E}_{p(c_1,\pi_{c_1},\tau_{c_1}^+)p(\tau_{c_1,2:K}^-)}\left[\log\left(\frac{K\cdot f_\psi(c_1,\pi_{c_1},\tau_{c_1}^+)}{f_\psi(c_1,\pi_{c_1},\tau_{c_1}^+)+\sum_{k=2}^K f_\psi(c_1,\pi_{c_1},\tau_{c_1,k}^-)}\right)\right] \\
&= \mathbb{E}_{p(c_1,\pi_{c_1})}\left[\mathbb{E}_{p(\tau_{c_1,2:K}^-)}\log\left(\frac{K\cdot f_\psi(c_1,\pi_{c_1},\tau_{c_1}^+)}{f_\psi(c_1,\pi_{c_1},\tau_{c_1}^+)+\sum_{k=2}^K f_\psi(c_1,\pi_{c_1},\tau_{c_1,k}^-)}\right)\right] \\
&\leq \log K
\end{aligned} \tag{9}
$$

Hence, we have $I_{\text{SaNCE}}(c;\pi_c;\tau_c|\psi,K) \leq \log K$ (similar to Equation 6).

Step 3. With the definition of K^*, we can prove $I_{\text{InfoNCE}}(c;\pi_c;\tau_c|\psi,K) \leq I_{\text{SaNCE}}(c;\pi_c;\tau_c|\psi,K)$ with the same sample size K. In task e_1 with context embedding c_1, SaNCE obtains positive and negative samples from the current task e_1. Hence the variable $c = c_1$ is constant, we have:

$$
\begin{aligned}
&I_{\text{SaNCE}}(c;\pi_c;\tau_c|\psi,K) \\
&= \mathbb{E}_{p(c_1,\pi_{c_1},\tau_{c_1}^+)p(\tau_{c_1,2:K}^-)}\left[\log\left(\frac{K\cdot f_\psi(c_1,\pi_{c_1},\tau_{c_1}^+)}{f_\psi(c_1,\pi_{c_1},\tau_{c_1}^+)+\sum_{k=2}^K f_\psi(c_1,\pi_{c_1},\tau_{c_1,k}^-)}\right)\right] \\
&\leq \mathbb{E}_{p(c_1,\pi_{c_1},\tau_{c_1}^+)p(\tau_{c_1,2:K^*_{\text{SaNCE}}}^-)}\left[\log\left(\frac{K^*_{\text{SaNCE}}\cdot f_\psi(c_1,\pi_{c_1},\tau_{c_1}^+)}{f_\psi(c_1,\pi_{c_1},\tau_{c_1}^+)+\sum_{k=2}^{K^*_{\text{SaNCE}}} f_\psi(c_1,\pi_{c_1},\tau_{c_1,k}^-)}\right)\right] \\
&= \mathbb{E}_{p(\pi_{c_1},\tau_{c_1}^+)p(\tau_{c_1,2:K^*_{\text{SaNCE}}}^-)}\left[\log\left(\frac{K^*_{\text{SaNCE}}\cdot f_\psi(\pi_{c_1},\tau_{c_1}^+)}{f_\psi(\pi_{c_1},\tau_{c_1}^+)+\sum_{k=2}^{K^*_{\text{SaNCE}}} f_\psi(\pi_{c_1},\tau_{c_1,k}^-)}\right)\right] \text{ (c_1 is constant.)} \\
&\approx \log K^*_{\text{SaNCE}}
\end{aligned} \tag{10}
$$

The required sample size $K^*_{\text{SaNCE}} = |c_1| \cdot |\pi| \cdot M = |\pi| \cdot M$. With $K \to K^*_{\text{SaNCE}}$, $I_{\text{SaNCE}}(c; \pi_c; \tau_c|\psi, K) \approx I_{\text{SaMI}}(c; \pi_c; \tau_c)$. Correspondingly, for InfoNCE, in the current task e_1 with context embedding c_1, positive samples are trajectories τ_1 generated after executing the skill π_1 in task e_1, and negative samples are trajectories $\{\tau^-_{c_2}, ...\}$ from other tasks $\{e_2, ...\}$. Under the definition of K^*, we have:

$$
\begin{aligned}
& I_{\text{InfoNCE}}(c; \pi_c; \tau_c|\psi, K) \\
&= \mathbb{E}_{p(c,\pi_c,\tau_{c_1})p(\tau_{c_{2:|c|}},{}_{2:K})} \left[\log \left(\frac{K \cdot f_\psi(c, \pi_c, \tau_{c_1})}{f_\psi(c, \pi_c, \tau_{c_1}) + \sum_{k=2}^{K} f_\psi(c, \pi_c, \tau_{c_{2:|c|},k})} \right) \right] \\
&\leq \mathbb{E}_{p(c,\pi_c,\tau_{c_1})p(\tau_{c_{2:|c|}},{}_{2:K^*_{\text{InfoNCE}}})} \left[\log \left(\frac{K^*_{\text{InfoNCE}} \cdot f_\psi(c, \pi_c, \tau_{c_1})}{f_\psi(c, \pi_c, \tau_{c_1}) + \sum_{k=2}^{K^*_{\text{InfoNCE}}} f_\psi(c, \pi_c, \tau_{c_{2:|c|},k})} \right) \right] \\
&\approx \log K^*_{\text{InfoNCE}},
\end{aligned}
\tag{11}
$$

where $K^*_{\text{InfoNCE}} = |c| \cdot |\pi| \cdot M \approx |c| \cdot K^*_{\text{SaNCE}}$. In real-world robotic control tasks, the sample space size significantly increases due to multiple environmental features $e = \{e^0, e^1, ..., e^N\}$. $|c|$ is the number of different tasks and increases exponentially due to the permutations and combinations of N environmental features. When current task e_1 has context embedding c_1, the $c_{2:|c|}$ refer to the context embedding for the other tasks. With $K \to K^*_{\text{InfoNCE}}$, $I_{\text{InfoNCE}}(c; \pi_c; \tau_c|\psi, K) \approx I_{\text{SaMI}}(c; \pi_c; \tau_c)$. Hence, during the training process, $I_{\text{InfoNCE}}(c; \pi_c; \tau_c|\psi, K) \leq I_{\text{SaNCE}}(c; \pi_c; \tau_c|\psi, K)$ with the same sample size K.

Step 4. What remains is to show $I_{\text{SaMI}}(c; \pi_c; \tau_c) \leq I(c; \tau_c)$. The MI for three variables is also called interaction information. According to the definition in McGill [1954], the SaMI can be presented as:

Proposition 2 *For the case of three variables, the MI could be written as* $I_{SaMI}(c; \pi_c; \tau_c) = I(c; \tau_c) - I(c; \tau_c|\pi_c)$.

Using this proposition, one can see that in the case of three variables, interaction information quantifies how much the information shared between two variables differs from what they share if the third variable is known. Several properties of interaction information in the case of three variables have been studied in the literature. Specifically, Yeung [1991] showed that

$$
-\min\{I(x; \tau_c|\pi_c), I(c; \pi_c|\tau_c), I(\tau_c; \pi_c|c)\} \leq I(c; \tau_c; \pi_c) \leq \min\{I(c; \tau_c), I(c; \pi_c), I(\tau_c; \pi_c)\}
\tag{12}
$$

Combining Equation 12 and Proposition 2, we have $I_{\text{SaMI}}(c; \pi_c; \tau_c) \leq I(c; \tau_c)$.

Putting it together. Hence, we have Lemma 2: we always have $I_{\text{InfoNCE}}(c; \pi_c; \tau_c|\psi, K) \leq I_{\text{SaNCE}}(c; \pi_c; \tau_c|\psi, K) \leq \log K \leq I_{\text{SaMI}}(c; \pi_c; \tau_c) \leq I(c; \tau_c)$ (shown in Figure 3), while learning a skill-aware context encoder ψ with SaNCE estimator. $K^*_{\text{SaNCE}} << K^*_{\text{InfoNCE}}$ so that $I_{\text{SaNCE}}(c; \pi_c; \tau_c|\psi, K)$ is a much tighter lower bound of the true $I_{\text{SaMI}(c;\pi_c;\tau_c)}$ than $I_{\text{InfoNCE}}(c; \pi_c; \tau_c|\psi, K)$.

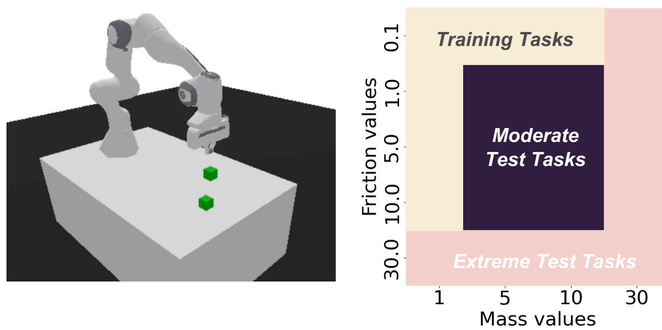
C Sample size of $I_{\text{Sa+InfoNCE}}$

We illustrate the sample size of $I_{\text{Sa+InfoNCE}}(c; \pi_c; \tau_c|\psi, K)$ in this section. Sa+InfoNCE plugs SaNCE into InfoNCE, and has positive samples $\tau^+_{c_1}$ from task e_1 after executing skill $\pi^+_{c_1}$, and negative samples are trajectories $\tau^-_{c_{1:K}}$ from executing skills $\pi^-_{c_{1:K}}$ in task $e_{1:K}$, respectively. Therefore, this is equivalent to observing the variable c, then observing the variable π_c, i.e., sampling from the distribution $p(\pi_c, \tau_c|c)p(c)$. We have:

$$
\begin{aligned}
& I_{\text{Sa+InfoNCE}}(c; \pi_c; \tau_c|\psi, K) \\
&= \mathbb{E}_{p(c)p(\pi^+_{c_1}, \tau^+_{c_1}|c)p\left(\left(\pi^-_{2:|c|}, \tau^-_{2:|c|}\right)_{2:K}\right)} \left[\log \left(\frac{K \cdot f_\psi(c, \pi^+_{c_1}, \tau^+_{c_1})}{f_\psi(c, \pi^+_{c_1}, \tau^+_{c_1}) + \sum_{k=2}^{K} f_\psi(c, \pi^-_{2:|c|,k}, \tau^-_{2:|c|,k})} \right) \right] \\
&\leq \mathbb{E}_{p(c)p(\pi^+_{c_1}, \tau^+_{c_1}|c)p\left(\left(\pi^-_{2:|c|}, \tau^-_{2:|c|}\right)_{2:K^*_{\text{Sa+InfoNCE}}}\right)} \left[\log \left(\frac{K^*_{\text{Sa+InfoNCE}} \cdot f_\psi(c, \pi^+_{c_1}, \tau^+_{c_1})}{f_\psi(c, \pi^+_{c_1}, \tau^+_{c_1}) + \sum_{k=2}^{K^*_{\text{Sa+InfoNCE}}} f_\psi(c, \pi^-_{2:|c|,k}, \tau^-_{2:|c|,k})} \right) \right] \\
&\approx \log K^*_{\text{Sa+InfoNCE}}
\end{aligned}
\tag{13}
$$

It should be noted that such a combination will increase the size of the negative sample space, i.e., $K^*_{\text{Sa+InfoNCE}} = \left(\sum_{i=1}^{|c|} |\pi^-_{c_i}| + |\pi^+_{c_1}|\right) \cdot M \geq K^*_{\text{SaNCE}}$. Figure 4 illustrates the differences in the size of the negative sample space, in which the negative sample space may vary across tasks (as shown by the non-lining-up bars in Figure 4 (c)). This is because we define negative samples as trajectories with low returns, so the size of the negative sample space is influenced by sampling randomness. With the same number K of samples, $I_{\text{Sa+InfoNCE}}$ is less precise and looser than I_{SaNCE}. Hence, a trade-off between sample diversity and the precision of the K-sample estimator is required.

D Environmental setup



(a) Panda-gym (b) Task setting

Figure 8: (a) Modified Panda-gym benchmarks, (b) the training tasks, the moderate test tasks and the extreme test tasks. The moderate test task setting just has a combinatorial interpolation. The extreme test task setting has unseen ranges of environmental features and is an extrapolation.

D.1 Modified Panda-gym

We modified the original Pick&Place task in Panda-gym [Gallouédec et al., 2021] by setting the z dimension (i.e., the desired height) of the cube's goal position equal to 0 [3] and maintaining the freedom [4] of grippers, so the agent can explore whether it is supposed to push the cube or grasp it. Skills in this benchmark are defined as:

- ***Pick&Place skill***: This skill specifically refers to the agent trying to use the gripper to grasp the cube, pick it up off the table and place it in the goal position. We determine the Pick&Place skill by detecting no contact points between the table and the cube, two contact points between the robot's end effector and the cube, and the cube's height being greater than half its width, meaning it is on the table.
- ***Push skill***: This skill refers to the agent giving a push force to the cube and then the cube slides to the goal position. We confirm the Push skill by detecting that the cube's height is equal to half its width.
- ***Other skills***: Besides the Pick&Place and Push skills, any other different modes of behaviour are classified as other skills. For example, with the Flip skill, the agent applies an initial force to the cube, causing it to roll on the surface or leave the table. In this case, we detect that the cube's height is greater than half its width, and the number of contact points between the cube and the table is greater than zero.

Some elements in the RL framework are defined as the following:

[3]If z is not equal to 0, Pick&Place skill is always needed to solve tasks.

[4]In the original Push task, the grippers are blocked to ensure the agent can only push cubes. However, this hinders the agent from learning Pick&Place skills, leading to failure when it encounters an "unpushable" scenario.

State space: we use feature vectors which contain cube position (3 dimensions), cube rotation (3 dimensions), cube velocity (3 dimensions), cube angular velocity (3 dimensions), end-effector position (3 dimensions), end-effector velocity (3 dimensions), gripper width (1 dimension), desired goal (3 dimensions) and achieved goal (3 dimensions). Environmental features are not included in the state.

Action space: the action space has 4 dimensions, the first three dimensions are the end-effector's position changes and the last dimension is the gripper's width change.

During training, we randomly select a combination of environmental features from a training set sampling combinations from sets: mass $= 1.0$ and friction $\in \{0.1, 1.0, 5.0, 10.0\}$; mass $\in \{1.0, 5.0, 10.0\}$ and friction $= 0.1$. At test time, we evaluate each algorithm in all tasks from the moderate test task setting, in which mass $\in \{5.0, 10.0\}$ and friction $\in \{1.0, 5.0, 10.0\}$ (shown in Figure 8(b)), and all tasks from extreme test task setting: mass $= 30.0$ and friction $\in \{0.1, 1.0, 5.0, 10.0, 30.0\}$; mass $\in \{1.0, 5.0, 10.0, 30.0\}$ and friction $= 30.0$ (shown in Figure 8(b)).

D.2 Modified MuJoCo

We extended the modified MuJoCo benchmark introduced in DOMINO [Mu et al., 2022] and CaDM [Lee et al., 2020]. Hence, in our extension, there are four new-added environments (Walker, Crippled Hopper, Crippled Walker, HumanoidStandup) compared with the original benchmark. Additionally, in our experiments, we used a different task set design (Table 3) than in the DOMINO and CaDM papers. For Hopper, Walker, Half-cheetah, Ant, HumanoidStandup and SlimHumanoid, we use the environments from the MuJoCo physics engine and use implementation available from Clavera et al. [2019b] and Seo et al. [2020], and scale the mass of every rigid link by scale factor m, and scale damping of every joint by scale factor d. For Crippled Ant, Crippled Hopper, Crippled Walker and Crippled Half-cheetah, we use implementation available from Seo et al. [2020] and scale the mass of every rigid link by scale factor m, scale damping of every joint by scale factor d, and randomly select one leg, and make it crippled to change the dynamic transitions. Generalisation performance is measured in two different regimes: moderate and extreme, where the moderate draws Environmental features from a closer range to the training range, compared to the extreme. We have provided our settings for training, extreme and moderate test tasks in Table 3.

D.3 Implementation details

In this section, we provide the implementation details for SaMI. We show the pseudo-code for using SaNCE during meta-training and meta-testing in Algorithm 1 and 2. Our codebase is built on top of the publicly released implementation Stable Baselines3 by Raffin et al. [2021] as well as the implementation of InfoNCE by Oord et al. [2019]. A public and open-source implementation of SaMI is available at https://github.com/uoe-agents/SaMI.

Base algorithm. We use SAC [Haarnoja et al., 2018] for downstream evaluation of the learned context embedding. SAC is an off-policy actor-critic method that uses the maximum entropy framework for

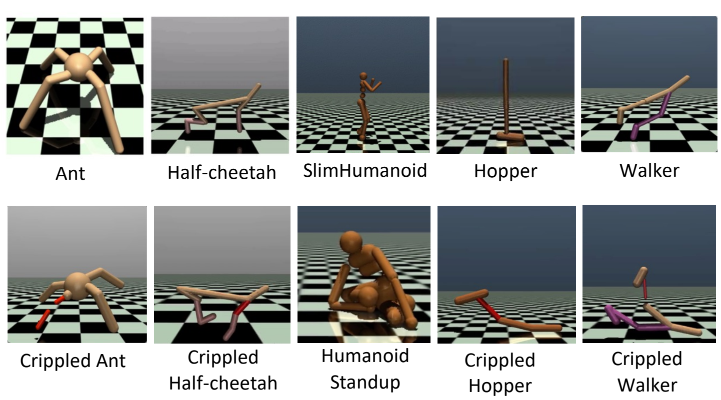


Figure 9: Ten environments in modified MuJoCo benchmark.

Table 3: Environmental features used for MuJoCo benchmark.

	Training	Test (Moderate)	Test (Extrame)	Episode Length
Half-cheetah	$m \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ $d \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$	$m \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$ $d \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$	1000
Ant	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$	$m \in \{0.20, 0.40, 1.60, 1.80\}$ $d \in \{0.20, 0.40, 1.60, 1.80\}$	$m \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$ $d \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$	1000
Hopper	$m \in \{0.75, 1.0, 1.25\}$ $d \in \{0.75, 1.0, 1.25\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$	$m \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$ $d \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$	1000
Crippled Hopper	$m \in \{0.75, 1.0, 1.25\}$ $d \in \{0.75, 1.0, 1.25\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$	$m \in \{0.20, 0.40, 1.60, 1.80, 4.0\}$ $d \in \{0.20, 0.40, 1.60, 1.80, 4.0\}$ Crippled Legs $R_1 \in \{0, 1, 2\}$	1000
SlimHumanoid	$m \in \{0.80, 0.90, 1.0, 1.15, 1.25\}$ $d \in \{0.80, 0.90, 1.0, 1.15, 1.25\}$	$m \in \{0.60, 0.70, 1.50, 1.60\}$ $d \in \{0.60, 0.70, 1.50, 1.60\}$	$m \in \{0.40, 0.50, 1.70, 1.80\}$ $d \in \{0.40, 0.50, 1.70, 1.80\}$	1000
HumanoidStandup	$m \in \{0.80, 0.90, 1.0, 1.15, 1.25\}$ $d \in \{0.80, 0.90, 1.0, 1.15, 1.25\}$	$m \in \{0.60, 0.70, 1.50, 1.60\}$ $d \in \{0.60, 0.70, 1.50, 1.60\}$	$m \in \{0.40, 0.50, 1.70, 1.80, 4.00\}$ $d \in \{0.40, 0.50, 1.70, 1.80, 4.00\}$	1000
Walker	$m \in \{0.75, 1.0, 1.25\}$ $d \in \{0.75, 1.0, 1.25\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$	$m \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$ $d \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$	2000
Crippled Walker	$m \in \{0.75, 1.0, 1.25\}$ $d \in \{0.75, 1.0, 1.25\}$ Crippled Joints (right leg) $= \{0, 1, 2\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$ Crippled Joints (right leg) $\in \{0, 1, 2\}$	$m \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$ $d \in \{0.20, 0.40, 1.60, 1.80, 4.00\}$ Crippled Joints (left leg) $\in \{3, 4, 5\}$	2000
Crippled Ant	$m \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ $d \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ Crippled Legs $R_1 \in \{0, 1, 2\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$ Crippled Legs $R_1 \in \{3\}$	$m \in \{0.20, 0.40, 1.60, 1.80\}$ $d \in \{0.20, 0.40, 1.60, 1.80\}$ Crippled Legs $R_1 \in \{3\}$	2000
Crippled Half-cheetah	$m \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ $d \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ Crippled Joints (front leg) $R_1 \in \{3, 4, 5\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$ Crippled Joints (back leg) $R_1 \in \{0, 1, 2\}$	$m \in \{0.20, 0.40, 1.60, 1.80\}$ $d \in \{0.20, 0.40, 1.60, 1.80\}$ Crippled Joints $\{R_1, R_2\} \in \{0, 1, 2, 3, 4, 5\}$ $(R_1 \neq R_2)$	2000

Table 4: Hyperparameters used in Panda-gym and MuJoCo benchmarks. Most hyperparameter values are unchanged across tasks except for contrastive batch size and SaNCE loss coefficient.

Hyperparameter	Value
Replay buffer size	100,000
Contrastive batch size	MuJoCo 12, Panda-gym 256
SaNCE loss coefficient α	MuJoCo 1.0, Panda-gym 0.01
Context embedding dimension	6
Hidden state dimension	128
Learning rate (actor, critic and encoder)	1e-3
Training frequency (actor, critic and encoder)	128
Gradient steps	16
Momentum context encoder ψ^* soft-update rate	0.05
SAC target soft-update rate	critic 0.01, actor 0.05
SAC batch size	256
Discount factor	0.99
Optimizer	Adam

soft policy iteration. At each iteration, SAC performs soft policy evaluation and improvement steps. We use the same SAC implementation across all baselines and other methods. In the Meta-training phase, we trained agents for 1.6 million timesteps in each environment on the Panda-gym and MuJoCo benchmarks. For meta-testing, we tested 100 episodes in each environment, with tasks randomly sampled from the moderate and extreme task sets.

Encoder architecture. For our method, the context encoder ψ is modelled as a Long Short-Term Memory (LSTM) that produces a 128-dimensional hidden state vector, subsequently processed through a single-layer feed-forward network to generate a 6-dimensional context embedding. We hope an agent can complete the three steps of "explore effectively, infer, adapt" within an episode. Therefore, we initialise the hidden state and cell state of LSTM to zero at the start of each episode. The actor and critic both use the same context encoder to embed trajectories. For contrastive learning, SaNCE utilises a momentum encoder ψ^* to generate positive and negative context embeddings [Laskin et al., 2020, He et al., 2020]. Formally, denoting the parameters of ψ as θ_ψ and those of ψ^* as θ_{ψ^*}, we update θ_{ψ^*} by:

$$\theta_{\psi^*} \leftarrow m \cdot \theta_\psi + (1 - m) \cdot \theta_{\psi^*}. \tag{14}$$

Here $m \in [0, 1)$ is a soft-update rate. Only the parameters θ_ψ are updated by back-propagation. The momentum update in Equation 14 makes θ_{ψ^*} evolve more smoothly by having them slowly track the θ_ψ with $m \ll 1$ (e.g., $m = 0.05$ in this research). This means that the target values are constrained to change slowly, greatly improving the stability of learning.

Hyperparameters. A full list of hyperparameters is displayed in Table 4.

Hardware. For each experiment run we use a single NVIDIA Volta V100 GPU with 32GB memory and a single CPU.

Algorithm 1 SaNCE Meta-training

Require: Batch of training tasks $\{e_n\}_{n=1,...,N}$ from $\xi_{train}(e)$, soft-update rate m;

Initialize RL replay buffer $\mathcal{B}_{RL}$, encoder replay buffer $\mathcal{B}_{enc}$;
Initialize parameters ψ for context encoder, ψ^* for momentum context encoder and ϕ for the off-policy SAC;
while not done **do**
  for each task e_n **do**
    for Roll-out time steps **do**
      for time step t < maximum episode length T **do**
        Update context embedding $c_n \sim \psi(c_n|\tau_{c_n,0:t})$
        Roll-out policy $\pi_{c_n}(a_t|s_t, c_n)$ and accumulate transition (s_t, a_t, r_t, s_{t+1});
      end for
      Add trajectory $\tau_{c_n} = \{s_0, a_0, r_0, s_1, r_1, ..., s_T, a_T, r_T\}$ to replay buffer $\mathcal{B}^n_{RL}$ and $\mathcal{B}^n_{enc}$;
    end for
  end for
  for each training step **do**
    for each task e_n **do**
      Sample RL batch $\{\tau_{c_n}\} \sim \mathcal{B}^n_{RL}$;
      Sample a positive sample $\tau^+_{c_n}$ for generating query with highest return, positive samples $\{\tau^-_{c_n}\}$ and negative samples $\{\tau^-_{c_n}\}$ for encoding positive and negative embeddings;
      Update ϕ with RL loss $\mathcal{L}_{\text{RL}}$;
      Update ψ with SaNCE loss $\mathcal{L}_{\text{SaNCE}}$ and RL loss $\mathcal{L}_{\text{RL}}$;
      $\theta_{\psi^*} \leftarrow m \cdot \theta_\psi + (1 - m) \cdot \theta_{\psi^*}$;
    end for
  end for
end while

Algorithm 2 SaNCE Meta-testing

Require: Batch of training tasks $\{e_n\}_{n=1,...,N}$ from $\xi_{test}(e)$;

while not done **do**
  for each task e_n **do**
    for each episode **do**
      for time step t < maximum episode length T **do**
        Update context embedding $c_n \sim \psi(c_n|\tau_{c_n,0:t})$
        Roll-out policy $\pi_{c_n}(a_t|s_t, c_n)$ and accumulate transition (s_t, a_t, r_t, s_{t+1});
      end for
    end for
  end for
end while

E Additional results

E.1 Balance contrastive and RL updates: loss coefficient α

While past work has learned hyperparameters to balance the contrastive loss coefficient α relative to the RL objective [Jaderberg et al., 2016, Bachman et al., 2019], we use both the contrastive and RL objectives together with equal weight $\alpha = 1.0$ to be optimal for the MuJoCo benchmark, and $\alpha = 0.01$ for the Panda-gym benchmark, we also analyse the effect of loss coefficient α for CCM, SaTESAC and SaCCM in MuJoCo (Figure 11) and Panda-gym (Figure 10) benchmarks.

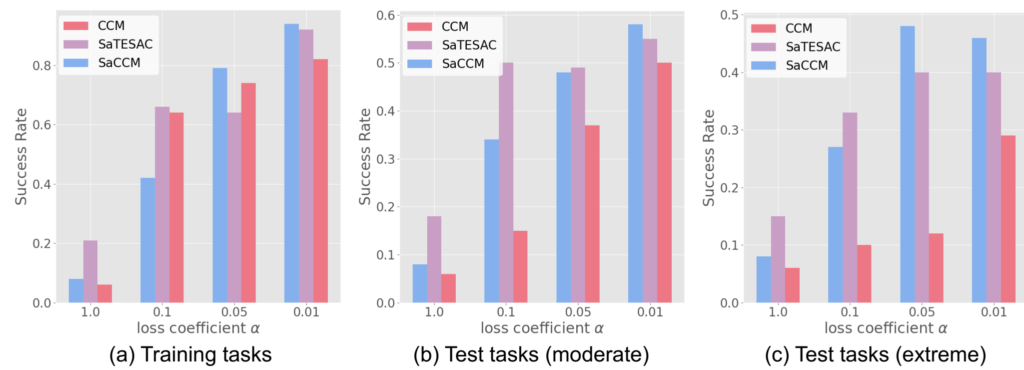


Figure 10: Loss coefficient α analysis of Panda-gym benchmark in training and test (moderate and extreme) tasks.

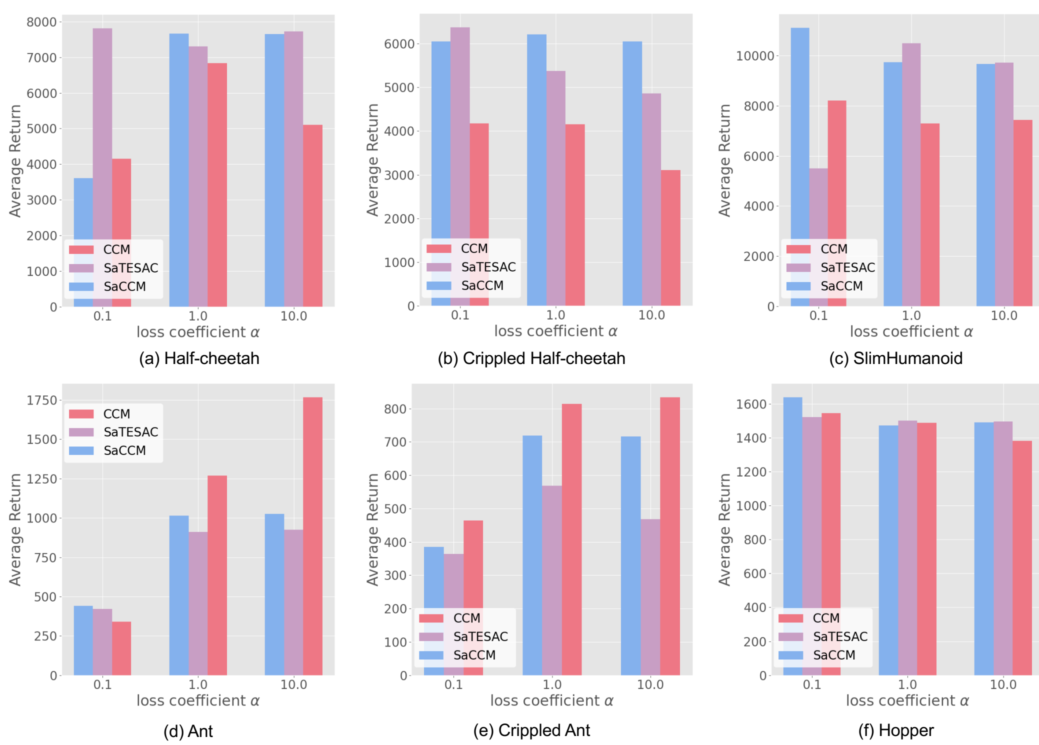
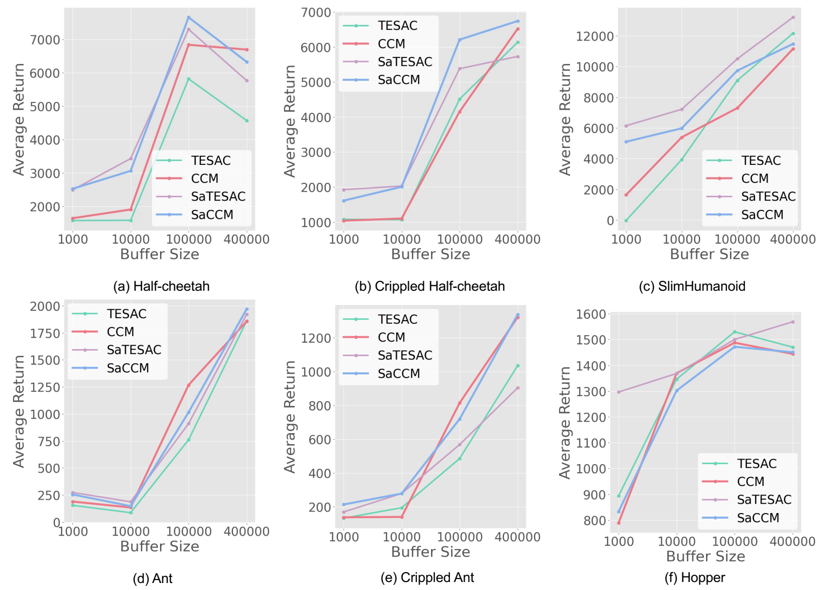


Figure 11: Loss coefficient α analysis of MuJoCo benchmark in training tasks.

E.2 Result of $\log$-K curse analysis

E.2.1 Buffer size

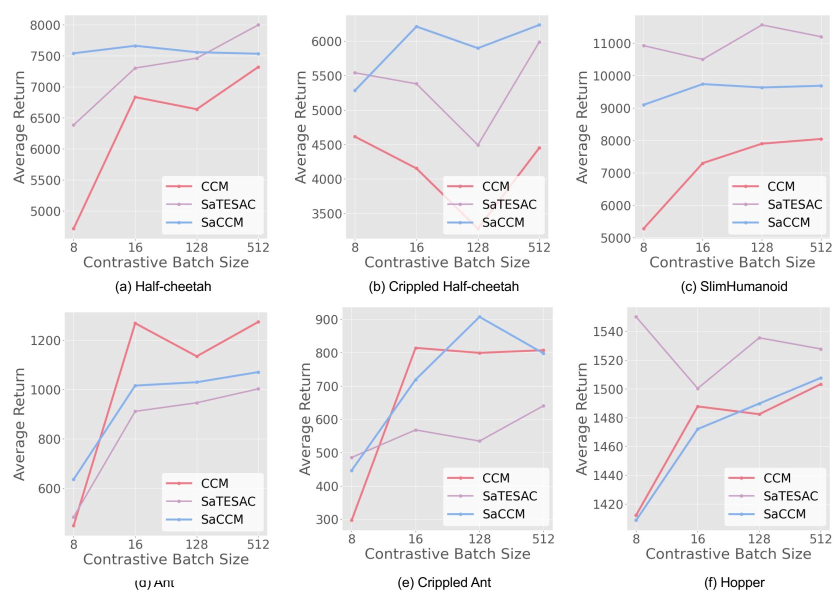


(a) Half-cheetah (b) Crippled Half-cheetah (c) SlimHumanoid

(d) Ant (e) Crippled Ant (f) Hopper

Figure 12: Comparison of different buffer sizes in MuJoCo benchmark in training tasks (over 5 seeds). Buffer size = 400000, 100000, 10000, and 1000.

E.2.2 contrastive batch size



(a) Half-cheetah (b) Crippled Half-cheetah (c) SlimHumanoid

(d) Ant (e) Crippled Ant (f) Hopper

Figure 13: Comparison of different contrastive batch sizes in MuJoCo benchmark in training tasks (over 5 seeds). contrastive batch size = 512, 128, 16, and 8.

F Further skill analysis

F.1 Panda-gym

F.1.1 Visualisation of context embedding

We visualise the context embedding via t-SNE [Van der Maaten and Hinton, 2008] (Figure 15) and PCA [Jolliffe, 2002] (Figure 14). When the mass of the cube is high (30 Kg and 10 Kg), the agent learned the Push skill (the yellow bounding box in Figure 1(a)), whereas with lower masses, the agent learned the Pick&Place skill. However, as shown in Figure 15(b), CCM did not display clear skill grouping. This indicates that SaMI extracts high-quality skill-related information from the trajectories and helps with embodying diverse skills.

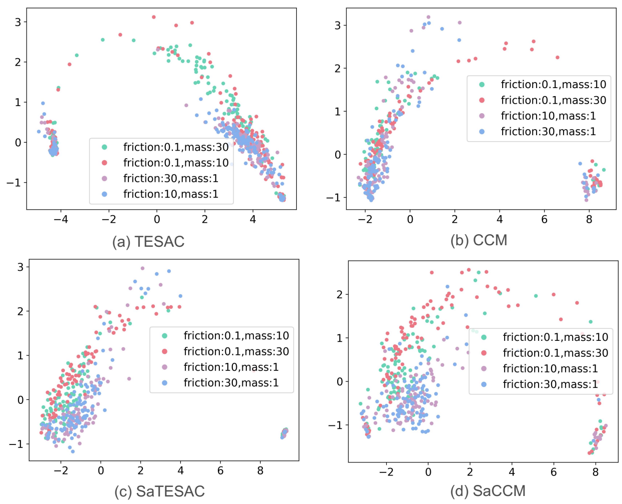


Figure 14: PCA visualization of context embedding extracted from trajectories collected in Panda-gym environments.

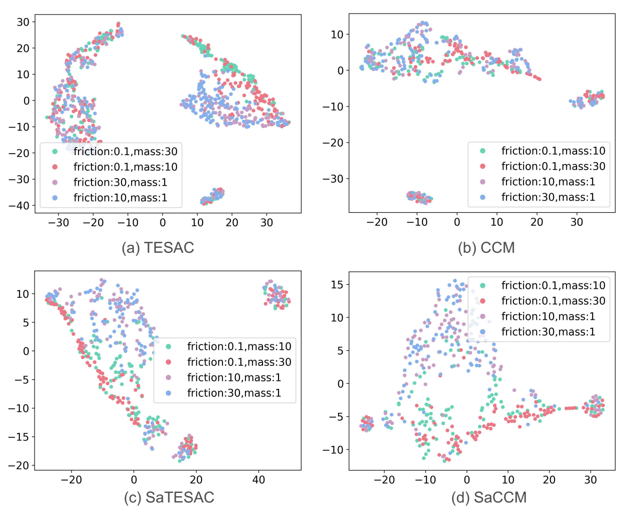
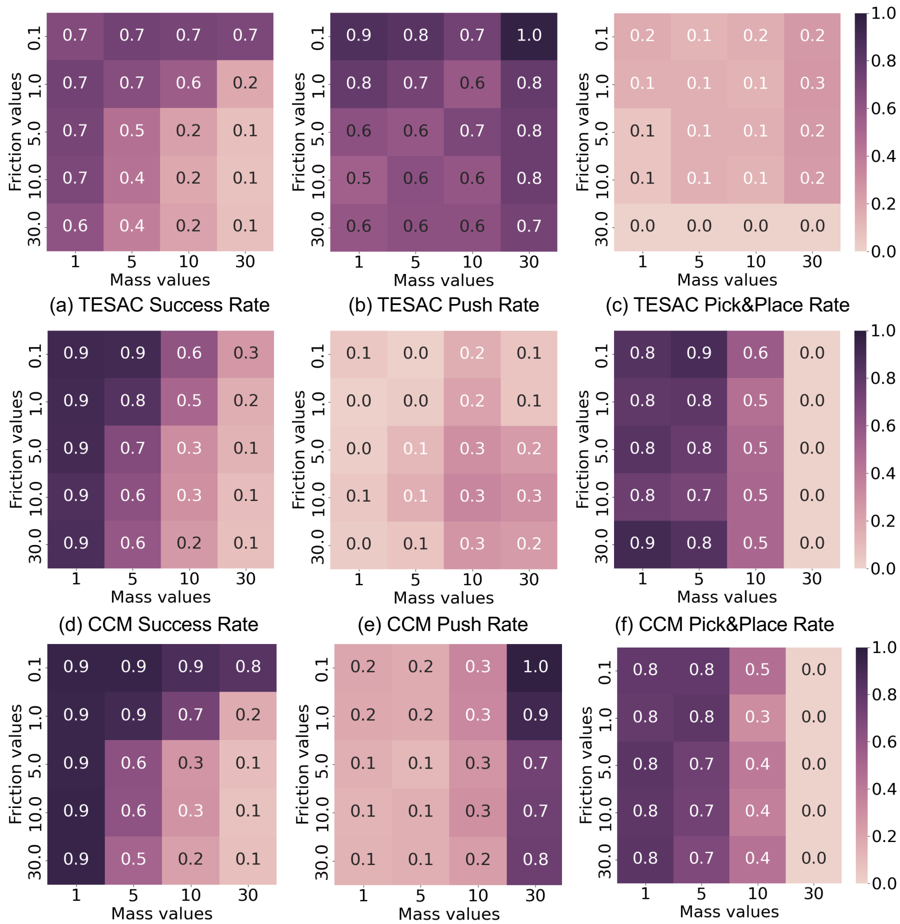


Figure 15: t-SNE visualization of context embedding extracted from trajectories collected in Panda-gym environments.

F.1.2 Heatmap of Panda-gym benchmark



(a) TESAC Success Rate (b) TESAC Push Rate (c) TESAC Pick&Place Rate

(d) CCM Success Rate (e) CCM Push Rate (f) CCM Pick&Place Rate

(g) SaTESAC Success Rate (h) SaTESAC Push Rate (i) SaTESAC Pick&Place Rate

Figure 16: Heatmap of success rate and learned skills of SaCCM. For each grid, the $(i, j)^{th}$ location shows the probability of the skills executed in 100 evaluations with $(\text{mass} = i, \text{friction} = j)$. We determine the skills by detecting contact points between the end effector and the cube and between the cube and the table.

This section provides the heatmap results and further analysis of TESAC, CCM, and SaTESAC on the Panda-gym benchmark. Initially, from the heatmap results of SaTESAC and SaCCM (Figure 6), we observed that with higher cube masses (30 and 10 kg), the agent adopted the Push skill (as indicated by the clustered points in the yellow bounding box in Figure 1(a)). At lower masses, the agent adopted the Pick&Place skill.

In contrast to CCM, as seen in Figures 16(e-f), CCM predominantly learned the Pick&Place skill, resulting in a decline in success rates for tasks at $mass = 30$, as the agent could not lift the cube off the table using the Pick&Place skill, as depicted in Figure 16(d). The visualisation of the context embedding (Figure 15) did not show clear grouping across different tasks.

Finally, TESAC only mastered the Push skill. The Push skill is relatively simpler to learn than the Pick&Place skill, as it does not require the agent to master manipulating fingers to pick up cubes. Consequently, TESAC's success rate notably decreased in environments with higher friction.

F.2 MuJoCo

We find that SaMI help the RL agents be versatile and embody multiple skills. For example, Figure 17 displays the t-SNE visualization results of four methods in the Crippled Half-cheetah environment for both test (moderate and extreme) and training tasks. Tasks corresponding to different crippled joints on the same leg are clustered together. Further, we have displayed rendered videos (https://github.com/uoe-agents/SaMI) to better demonstrate the different skills learned.

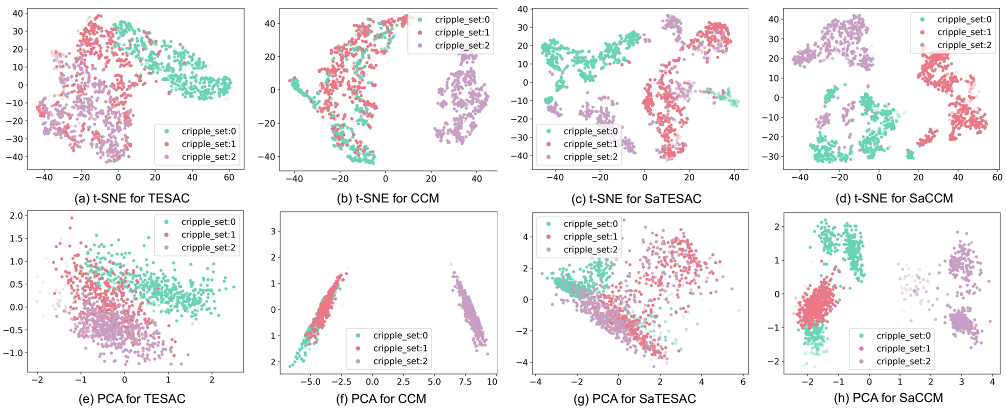


Figure 17: t-SNE and PCA visualisation of context embedding extracted from trajectories collected in Crippled Half-cheetah environment.

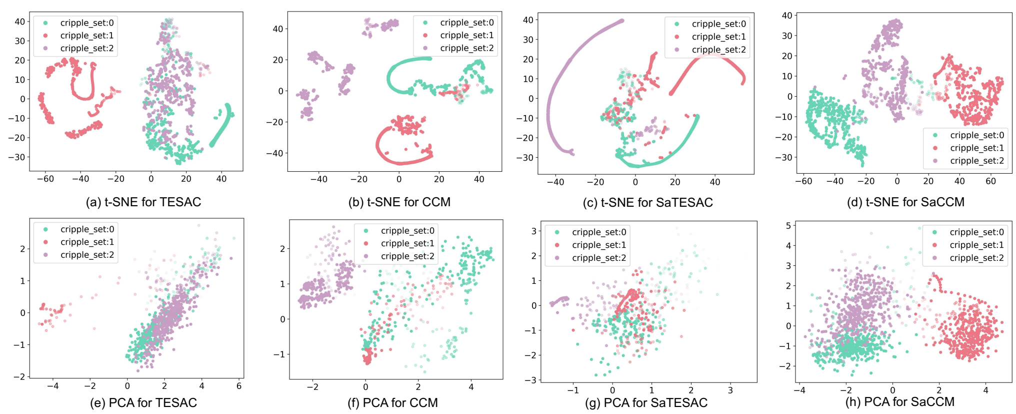


Figure 18: t-SNE and PCA visualization of context embedding extracted from trajectories collected in Crippled Ant environment.

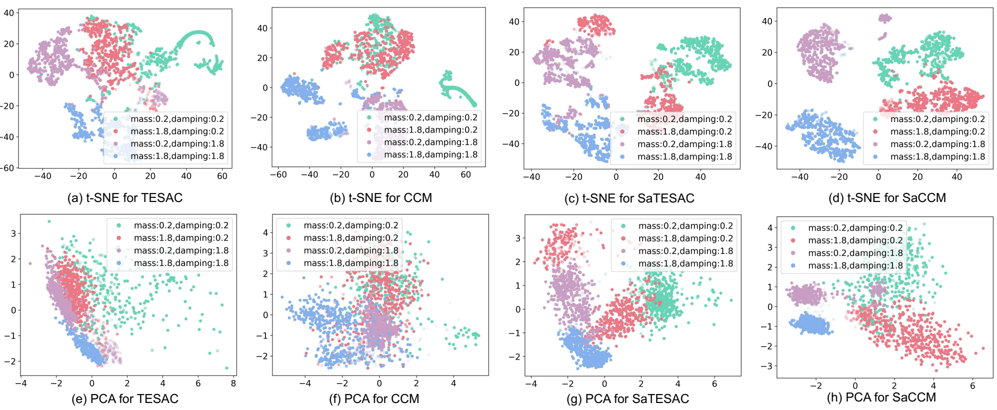


Figure 19: t-SNE and PCA visualization of context embedding extracted from trajectories collected in Half-cheetah environment.

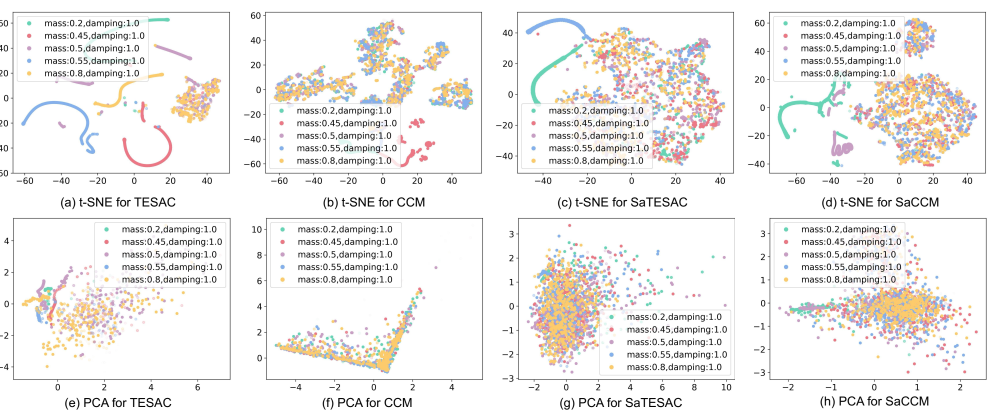


Figure 20: t-SNE and PCA visualization of context embedding extracted from trajectories collected in Ant environment.

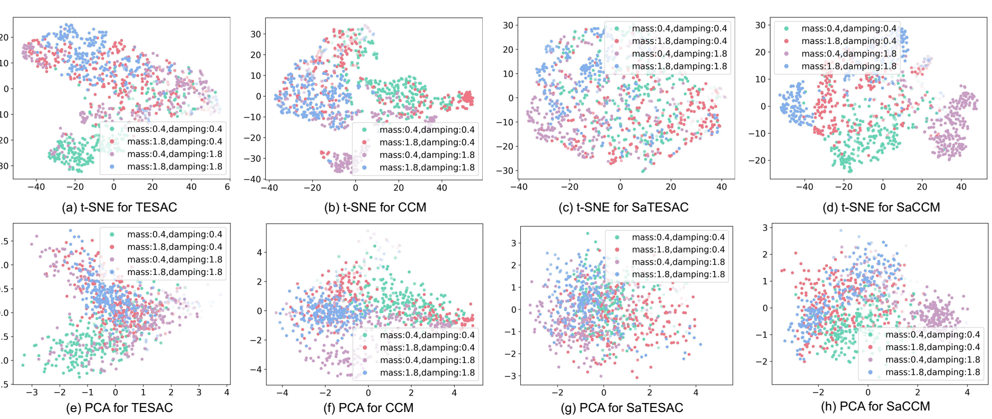


Figure 21: t-SNE and PCA visualization of context embedding extracted from trajectories collected in SlimHumanoid environment.

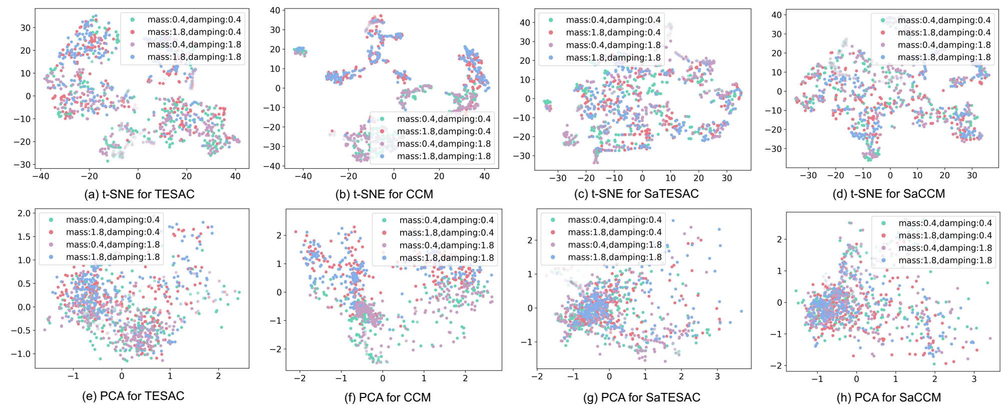


Figure 22: t-SNE and PCA visualization of context embedding extracted from trajectories collected in Hopper environment.

Table 5: Environmental features used for MuJoCo benchmark from DOMINO and CaDM.

	Training	Test (Moderate)	Test (Extrame)	Episode Length
Half-cheetah	$m \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ $d \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$	$m \in \{0.20, 0.40, 1.60, 1.80\}$ $d \in \{0.20, 0.40, 1.60, 1.80\}$	1000
Ant	$m \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ $d \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$	$m \in \{0.20, 0.40, 1.60, 1.80\}$ $d \in \{0.20, 0.40, 1.60, 1.80\}$	1000
Hopper	$m \in \{0.75, 1.0, 1.25\}$ $d \in \{0.75, 1.0, 1.25\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$	$m \in \{0.20, 0.40, 1.60, 1.80\}$ $d \in \{0.20, 0.40, 1.60, 1.80\}$	500
SlimHumanoid	$m \in \{0.80, 0.90, 1.0, 1.15, 1.25\}$ $d \in \{0.80, 0.90, 1.0, 1.15, 1.25\}$	$m \in \{0.60, 0.70, 1.50, 1.60\}$ $d \in \{0.60, 0.70, 1.50, 1.60\}$	$m \in \{0.40, 0.50, 1.70, 1.80\}$ $d \in \{0.40, 0.50, 1.70, 1.80\}$	500
Crippled Ant	$m \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ $d \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ Crippled Joints: $\{0, 1, 2\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$ Crippled Joints: $\{3\}$	$m \in \{0.20, 0.40, 1.60, 1.80\}$ $d \in \{0.20, 0.40, 1.60, 1.80\}$ Crippled Joints: $\{3\}$	1000
Crippled Half-cheetah	$m \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ $d \in \{0.75, 0.85, 1.0, 1.15, 1.25\}$ Crippled Joints: $\{0, 1, 2, 3\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$ Crippled Joints: $\{4, 5\}$	$m \in \{0.20, 0.40, 1.60, 1.80\}$ $d \in \{0.20, 0.40, 1.60, 1.80\}$ Crippled Joints: $\{4, 5\}$	1000

Table 6: Comparison of average return $\pm$ standard deviation with baselines in MuJoCo benchmark (over 5 seeds). The **bold text** signifies the highest average return. The numerical results for PPO+DOMINO are copied from Mu et al. [2022]; the numerical results for PPO+CaDM, Vanilla+CaDM, and PE-TS+CaDM are copied from Lee et al. [2020].

	Ant			Half-cheetah		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
PPO+DOMINO	227±86	216±52		2472±803	1034±476	
PPO+CaDM	268.6±77.0	228.8±48.4	199.2±52.1	2652.0±1133.6	1224.2±630.0	1021.1±676.6
Vanilla+CaDM	1851.0±113.7	1315.7±45.5	821.4±113.5	3536.5±641.7	1556.1±260.6	1264.5±228.7
PE-TS+CaDM	**2848.4±61.9**	**2121.0±60.4**	**1200.7±21.8**	**8264.0±1374.0**	**7087.2±1495.6**	**4661.8±783.9**
SaTESAC	908±65	640±117	532±88	7430±1026	4058±890	1780±102
SaCCM	928±141	635±94	555±88	7154±965	3849±689	1926±218

	SlimHumanoid			Crippled Half-Cheetah		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
PPO+DOMINO	7825±1256	5258±1039		2503±658	1326±491	
PPO+CaDM	10455.0±1004.9	4975.7±1305.7	3015.1±1508.3	2356.6±624.3	1454.0±462.6	1025.0±296.2
Vanilla+CaDM	1758.2±459.1	1228.9±374.0	1487.9±339.0	2435.1±880.4	1375.3±290.6	966.9±89.4
PE-TS+CaDM	1371.9±400.0	903.7±343.9	814.5±274.8	3294.9±733.9	2618.7±647.1	1294.2±214.9
SaTESAC	**10216±1620**	**7886±2203**	**6123±1403**	**5169±730**	2184±592	**1628±281**
SaCCM	**9312±705**	**7430±1587**	**6473±2001**	**5709±744**	2795±446	**2115±466**

G A comparison with DOMINO and CaDM

In this section, we give a brief comparison between our methods and methods from DOMINO [Mu et al., 2022] and CaDM [Lee et al., 2020] in the MuJoCo benchmark because we are using the exact same environmental setting (shown in Table 5).

DOMINO [Mu et al., 2022] is based on the InfoNCE K-sample estimator. Their implementation, PPO+DOMINO, is a model-free RL algorithm with a pre-trained context encoder. This encoder reduces the demand for large contrastive batch sizes during training by decoupling representation learning for each modality, simplifying tasks while leveraging shared information. However, a pre-trained encoder necessitates a large sample volume, with DOMINO training PPO agents for 5 million timesteps on the MuJoCo benchmark. In contrast, SaTESAC and SaCCM, trained for 1.6 million timesteps without pre-trained encoders, achieve considerably higher average returns across four environments (Table 6). Therefore, it is crucial to focus on extracting mutual information in contrastive learning that directly optimises downstream tasks, integrating rather than segregating representation learning from task performance.

CaDM [Lee et al., 2020] proposes a context-aware dynamics model adaptable to changes in dynamics. Specifically, they utilise contrastive learning to learn context embeddings, and then predict the next state conditioned on them. We copy the numerical results of PPO+CaDM, Vanilla+CaDM, and PE-TS+CaDM from CaDM [Lee et al., 2020] as their environmental setting is identical to ours, where PPO+CaDM is a model-free RL algorithm, while Vanilla+CaDM and PE-TS+CaDM are model-based. The model-free RL approach, PPO+CaDM, is trained for 5 million timesteps on the MuJoCo benchmark. As shown in Table 6, SaTESAC and SaCCM significantly outperform PPO+CaDM. The model-based RL algorithms, Vanilla+CaDM and PE-TS+CaDM, require 2 million timesteps for learning in model-based setups, compared to our fewer samples (i.e., million timesteps). In the Ant environment, Vanilla+CaDM and PE-TS+CaDM achieve higher returns than SaTESAC and SaCCM; similarly, in the Half-cheetah environment, PE-TS+CaDM outperforms them. Results in the SlimHumanoid and Crippled Half-cheetah environments show that skill-aware context embeddings are notably effective. An insight here is that our method outperforms the model-free CaCM approach, but not the model-based one. This is consistent with what is empirically found in CaDM [Lee et al., 2020]: prediction models are more effective when the transition function changes across tasks. Therefore, we consider that a model-based approach to SaMI could be an interesting extension for future work.

Table 7: The p-value of the statistical hypothesis tests (t-tests) for comparing the effectiveness of SaMI in MuJoCo benchmark (over 5 seeds). $*$ next to the number means that the algorithm with SaMI has statistically significant improvement over the same algorithm without SaMI at a significance level of 0.05. The "SaTESAC-TESAC" row indicates the p-value for the return improvement brought by SaMI to TESAC; the "SaCCM-CCM" row indicates the p-value for the return improvement brought by SaMI to CCM.

	Crippled Ant			Crippled Half-cheetah		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
SaTESAC-TESAC	0.121	0.109	9.54E-07*	0.154	0.0024*	0.0889
SaCCM-CCM	0.913	0.108	0.008*	0.04*	0.307	0.106
	Ant			**Half-cheetah**		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
SaTESAC-TESAC	0.136	0.034*	0.021*	0.346	0.224	0.004*
SaCCM-CCM	0.138	0.275	0.163	0.73	0.791	0.005*
	SlimHumanoid			**HumanoidStandup**		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
SaTESAC-TESAC	0.139	0.456	0.006*	0.037*	0.129	0.003*
SaCCM-CCM	0.113	0.059	0.008*	0.048	0.027*	0.01
	Hopper			**Crippled Hopper**		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
SaTESAC-TESAC	0.747	0.089	0.707	0.459	0.69	0.088
SaCCM-CCM	0.52	0.599	0.969	0.967	0.897	0.0002*
	Walker			**Crippled Walker**		
	Training	Test (moderate)	Test (extreme)	Training	Test (moderate)	Test (extreme)
SaTESAC-TESAC	0.312	0.082	0.003*	0.223	0.048*	0.028*
SaCCM-CCM	0.55	0.541	0.022*	0.794	0.079	0.011*

Table 8: The p-value of the statistical hypothesis tests (t-tests) for comparing the effectiveness of SaMI in Panda-gym benchmark (over 5 seeds). $*$ next to the number means that the algorithm with SaMI has statistically significant improvement over the same algorithm without SaMI at a significance level of 0.05. The "SaTESAC-TESAC" row indicates the p-value for the return improvement brought by SaMI to TESAC; the "SaCCM-CCM" row indicates the p-value for the return improvement brought by SaMI to CCM.

	Training	Test (moderate)	Test (extreme)
SaTESAC-TESAC	0.000185*	0.000119*	0.001225*
SaCCM-CCM	0.000200*	0.002900*	0.000700*

H Statistical hypothesis tests (t-tests)

We used a t-test [Rice and Rice, 2007] to conduct a statistical hypothesis test to determine whether SaMI brought a statistically significant improvement. we reported the p-value of the t-test in MuJoCo (Table 7) and Panda-gym (Table 8) benchmarks. $*$ next to the number is used to indicate that the algorithm with SaMI has statistically significant improvement over the same algorithm without SaMI at a significance level of 0.05. From Table 7 and 8, SaMI brings significant improvement on the extreme test set in which the RL agent needs to execute diverse skills. The statistically significant test aligns with our results in the skill analysis (i.e., video demos, t-SNE and PCA visualisation). In complex environments that require high skill diversity from the RL agent, the statistically significant improvement and higher returns/success rates brought by SaMI are evident.