Skill-aware Mutual Information Optimisation for Generalisation in Reinforcement Learning

Xuehui Yu[1,2] **Mhairi Dunion**[2] **Xin Li**[1] **Stefano V. Albrecht**[2]
[1]Harbin Institute of Technology [2]University of Edinburgh
{yuxuehui,22s103169}@stu.hit.edu.cn
{mhairi.dunion,s.albrecht}@ed.ac.uk

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 behaviour). 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 trained with SaNCE demonstrates greater robustness to a reduction in the number of available samples, thus possessing the potential to overcome the $\log$-K curse.

1 Introduction

Reinforcement Learning (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 is required to 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

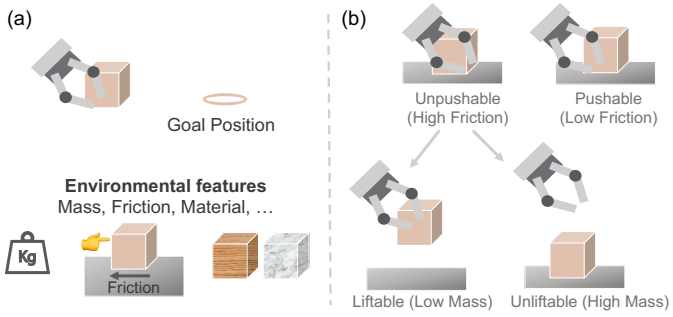


Figure 1: (a) In a cube-moving environment, tasks are defined according to different environmental features. (b) Different tasks have different transition dynamics caused by underlying environmental features, hence optimal skills are different across tasks.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

the cube to avoid friction, rather than continuing to push. Recent advances in Meta-Reinforcement Learning (Meta-RL) [Lee et al., 2020, Agarwal et al., 2021, Mu et al., 2022, Dunion et al., 2023b,a, McInroe et al., 2024] enable agents to understand environmental features by inferring context embeddings from a small amount of exploration, and to train a policy conditioned on the context embedding to generalise to novel tasks.

In recent years, unsupervised contrastive learning algorithms have been shown to learn context embeddings that perform remarkably well on generalisation tasks [Clavera et al., 2019a, Lee et al., 2020]. In particular, some Meta-RL algorithms [Fu et al., 2021, Wang et al., 2021, Li et al., 2021, Sang et al., 2022] train context encoders by maximising InfoNCE [Oord et al., 2019], which has yielded vital insights into contrastive learning through the lens of mutual information (MI) analysis. The InfoNCE objective can be interpreted as a K-sample lower bound on the MI between trajectories and context embeddings. Despite significant advances, 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) Existing *K*-sample MI estimators are sensitive to the sample size *K* (i.e., the log-*K* curse)** [Poole et al., 2019]. The low sample efficiency of RL [Franke et al., 2021] and the sample limitations in zero-shot generalisation make collecting a substantial quantity of samples often impractical [Arora et al., 2019, Nozawa and Sato, 2021]. The effectiveness of K-sample MI estimators breaks down with a finite 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. This also reduces the necessary sample size by focusing solely on skill-related information. In this work, (1) we introduce ***Skill-aware Mutual Information (SaMI)***, a generalised form of MI objective between context embeddings, skills, and trajectories, designed to address issue (i). We provide a theoretical proof showing that by introducing skills as a third variable into the MI of context embeddings and trajectories, the resulting SaMI is smaller and easier to optimise. Furthermore, (2) we propose a data-efficient K-sample estimator, ***Skill-aware Noise Contrastive Estimation (SaNCE)*** to optimise SaMI, effectively addressing issue (ii). Additionally, (3) we propose a practical skill-aware trajectory sampling strategy that shows how to sample positive and negative examples without relying on any prior skill distribution. In that way, Meta-RL agents autonomously acquire a set of skills applicable to many tasks, with these skills emerging solely from the SaMI learning objective and data.

We demonstrate empirically in MuJoCo [Todorov et al., 2012] and Panda-gym [Gallouédec et al., 2021] that SaMI enhances the zero-shot generalisation capabilities of two Meta-RL algorithms [Yu et al., 2020, Fu et al., 2021] by achieving higher returns and success rates on previously unseen tasks, ranging from moderate to extreme difficulty. Visualisation of the learned context embeddings reveals distinct clusters corresponding to different skills, suggesting that the SaMI learning objective enables the context encoder to capture skill-related information from trajectories and incentivise Meta-RL agents to acquire a diverse set of skills. Moreover, SaNCE enables Meta-RL algorithms to use smaller sample spaces while achieving improved downstream control performance, indicating their potential to overcome the $\log$-K curse.

2 Related works

Meta-RL. By conditioning on an effective context embedding, Meta-RL policies can zero-shot generalise to new tasks with a small amount of exploration Kirk et al. [2023]. 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 with task embeddings as inputs to policies (TESAC) [Hausman et al., 2018, Yu et al., 2020] parameterises the learned policies through a shared embedding space, aiming to maximise the average returns across tasks. However, the update signals from the RL loss are stochastic and weak, and may not capture the similarity relations among tasks [Fu et al.,

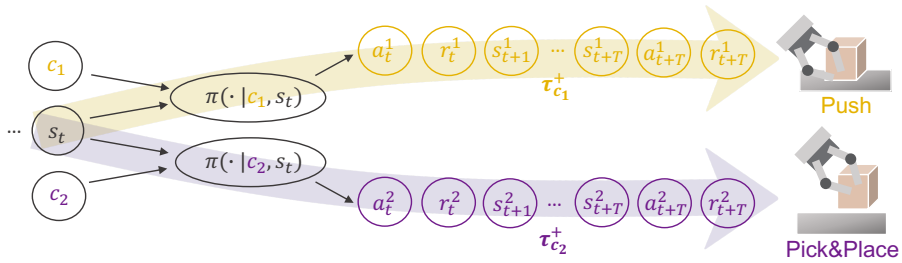


Figure 2: A policy π conditioned on a fixed context embedding c is defined as a skill $\pi(\cdot|c)$ (shortened as π_c). The policy π conditioned on a fixed c alters the state of the environment in a consistent way, thereby exhibiting a mode of skill. The skill $\pi(\cdot|c_1)$ moves the cube on the 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 $\tau_{c_2}^+$.

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 MI estimation [Poole et al., 2019]. MI is often estimated using InfoNCE [Oord et al., 2019] that has gained recent attention due to its lower variance [Song and Ermon, 2020] and superior performance in downstream tasks. However, InfoNCE may underestimate the true MI when the sample size K is finite. To address this limitation, CCM [Fu et al., 2021] uses a large number of samples for maximising InfoNCE. DOMINO [Mu et al., 2022] reduces the true MI by introducing an independence assumption; however, this results in biased estimates. 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 sampled from $\xi_{\text{test}}(e)$.

Contrastive learning. In Meta-RL, the context encoder $\psi(c|\tau_{c,0:t})$ 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 [Fu et al., 2021]. Then, the policy π, conditioned on context embedding c, consumes the current state s_t and outputs the action a_t. As a key component, the quality of context embedding c can affect algorithms' performance significantly. MI is an effective measure of embedding quality [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 on 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 $f_\psi(x,y) = e^{\psi(x)^\top \cdot \psi(y)/\beta}$, 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 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 (as shown in Figure 2). While an unbounded sample size K for learning effective context embeddings is theoretically feasible and assumed in many studies Mu et al. [2022], Lee et al. [2020], it is often impractical in practice. Therefore, building on Lemma 1, we derive two key insights with limited samples: (1) generalising the current MI objective to be smaller than $I(x;y)$ (see Section 4.2); (2) developing a K-sample estimator tighter than I_{InfoNCE} (see Section 4.3).

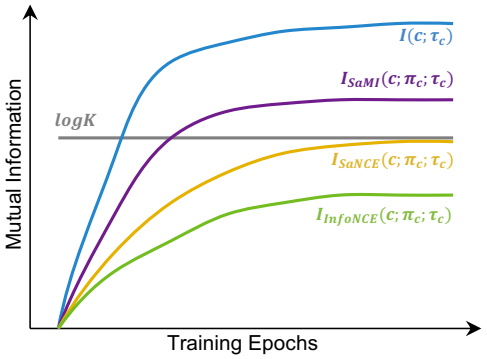


Figure 3: $I_{\text{InfoNCE}(c;\pi_c;\tau_c)}$, with a finite sample size of K, is a loose lower bound of $I(c;\tau_c)$ and leads to lower performance embeddings. $I_{\text{SaMI}}(c;\pi_c;\tau_c)$ is a lower ground-truth MI, and $I_{\text{SaNCE}}(c;\pi_c;\tau_c)$ is a tighter lower bound.

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

We aim for our MI learning objective to incentivise agents to acquire a diverse set of skills, enabling them to generalise effectively across tasks. 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 a limited amount of exploration, an agent should be able to infer the task (i.e., environmental features $e = e^0, e^1, \ldots, e^N$) and adapt accordingly within the current episode. The context em-

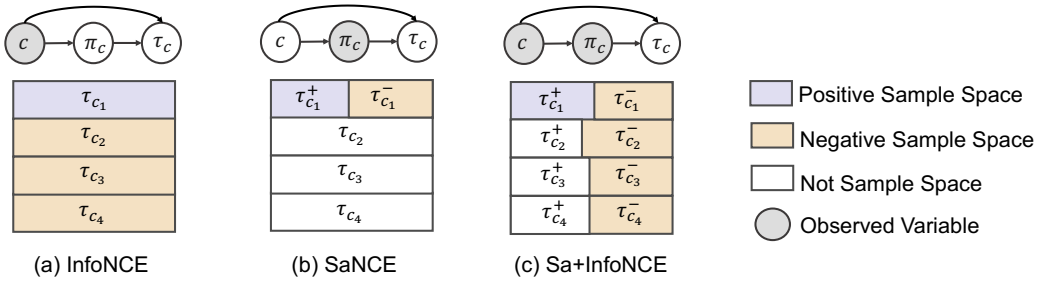


Figure 4: A comparison of sample spaces for task e_1. Positive samples τ_{c_1} or $\tau^+_{c_1}$ are always from current task e_1. For SaNCE, in a task e_k with embedding c_k, the positive skill $\pi^+_{c_k}$ conditions on c_k and generates positive trajectories $\tau^+_{c_k}$, and the negative skill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

bedding should encompass skill-related information, guiding the policy on when to explore new skills or switch between existing ones. We propose that the context encoder ψ should be trained by maximising the MI between the context embedding c, skills π_c, and trajectories τ_c. To this end, we propose a novel MI optimisation objective, **Skill-aware Mutual Information (SaMI)**, defined as:

$$I_{\text{SaMI}}(c;\pi_c;\tau_c) = I(c;\tau_c) - I(c;\tau_c|\pi_c). \tag{2}$$

SaMI is defined according to interaction information [McGill, 1954], serving as a generalisation of MI for three variables $\{c, \pi_c, \tau_c\}$. Although we cannot evaluate $p(c, \pi_c, \tau_c)$ directly, we approximate it by Monte-Carlo sampling, using K samples from $p(c, \pi_c, \tau_c)$. As illustrated in Figure 3, a context encoder ψ trained with the objective of maximising $I_{\text{SaMI}}(c;\pi_c;\tau_c)$ converges more quickly, as $I_{\text{SaMI}}(c;\pi_c;\tau_c) \leq I(c;\tau_c)$ (see proof in Appendix B). By focusing more on skill-related information, I_{SaMI} enables agents to autonomously discover a diverse range of skills for handling multiple tasks.

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. To address this, we propose a novel K-sample estimator that requires a significantly smaller sample size $K \ll +\infty$. 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.*

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 $K^*_{\text{SaNCE}} < K^*_{\text{InfoNCE}}$. 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

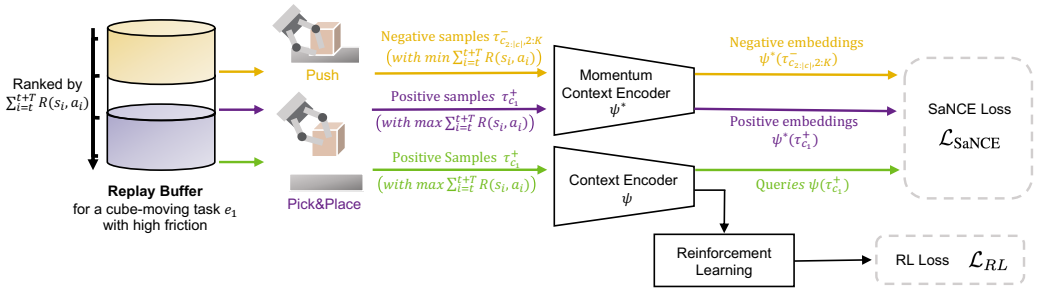


Figure 5: A practical framework for using SaNCE in the meta-training phase. During meta-training, we sample trajectories from the replay buffer for off-policy training. Queries are generated by a context encoder ψ, which is updated with gradients from both the SaNCE loss $\mathcal{L}_{\text{SaNCE}}$ 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.

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}:

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

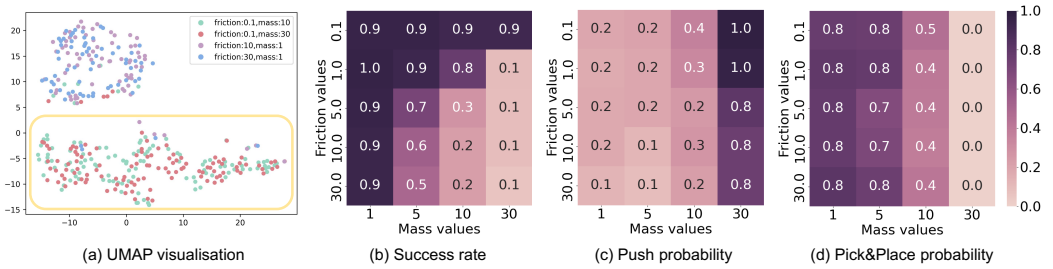
where $f_\psi(c_1, \pi_{c_1}, \tau_{c_1}) = e^{\psi(\tau_{c_1})^\top \cdot \psi^*(\tau_{c_1})/\beta}$. The query $c_1 = \psi(\tau_{c_1})$ is generated by the context encoder ψ. For training stability, we use a momentum encoder ψ^* to produce the positive and negative embeddings. SaNCE significantly reduces 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 can be used alone or combined with other optimisation objectives to train context encoders in Meta-RL algorithms. For instance, integrating SaNCE with InfoNCE diversifies the negative sample space, with $K^*_{\text{Sa+InfoNCE}} = \left(\sum_{i=1}^{|c|} |\pi^-_{c_i}| + |\pi^+_{c_1}| \right) \cdot M$. The sample space for $I_{\text{Sa+InfoNCE}}$ is depicted in Figure 4(c) and further analysed in detail in Appendix C.

4.4 Skill-aware trajectory sampling strategy

In this section, we propose a practical trajectory sampling method. Methods focusing on skill diversity often rely heavily on accurately defining and identifying individual skills [Eysenbach et al., 2018]. Some of these methods require a prior skill distribution, which is often inaccessible [Shi et al., 2022], and it is impractical to enumerate all possible skills that we hope the model to learn. Besides, 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. Consequently, we do not directly teach any of these skills or assume any prior skill distribution. Diverse skills naturally emerge from the incentives of the SaMI learning objective in a multi-task setting, driven by the inherent need to develop generalisable skills. For example, in high-friction tasks, the agent must acquire the Pick&Place skill to avoid large frictional forces, whereas in high-mass tasks, the agent must learn the Push skill since it cannot lift the cube. In each task, we only identify whether the skills



(a) UMAP visualisation (b) Success rate (c) Push probability (d) Pick&Place probability

Figure 6: (a) UMAP visualisation of context embeddings for the SaCCM in the Panda-gym environment, with points in the yellow box representing the Push skill in high-mass tasks. Heatmap of (b) success rate, (c) Push skill probability, and (d) Pick&Place skill probability for SaCCM. In large-mass scenarios, the Push skill is more likely to be executed than Pick&Place.

are optimal; in this way, under a multi-task setting, the agent will acquire a set of general skills that are applicable to many tasks.

In a given task e, *positive skills* π_c^+ are defined as optimal skills achieving highest return $\sum_{i=t}^{t+T} R(s_i, a_i)$, whereas *negative skills* π_c^- are those that result in lower returns. As a result, the positive sample τ_c^+ consists of trajectories generated by the skills with the highest ranked returns, while the negative samples correspond to those with the lowest returns. This straightforward approach of selecting positive samples based on the ranked highest return effectively aligns with positive skills and mitigates the challenge of hard negative examples [Robinson et al., 2021]. The SaNCE loss is then minimised to bring the context embeddings of the highest return trajectories closer while distancing those of negative trajectories. By the end of training, the top-ranked trajectories in the ranked replay buffer correspond to positive samples τ_c^+ with high returns, while the lower-ranked trajectories represent negative samples τ_c^- with low returns. However, at the beginning of training, it is likely that all trajectories have low returns. Therefore, our SaNCE loss is a soft variant of the K-sample SaNCE:

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

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

Our experiments aim to answer the following 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 evaluate our method on 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, which are sampled at the start of each episode during meta-training and meta-testing. During meta-training, we uniform-randomly select a combination of environmental features from a training task set. At test time, we evaluate each algorithm in unseen tasks with 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. Our results report the mean and standard deviation of models trained over five seeds in both training and test tasks. Further details are available in Appendix D.

[1] Our modified benchmarks are open-sourced at https://github.com/uoe-agents/Skill-aware-Panda-gym

Baselines. The loss function of the context encoder in all algorithms consists of two key components: the RL loss and the contrastive loss. The RL loss $\mathcal{L}_{RL}$, which is the same across all methods, corresponds to the RL value function loss. Our primary comparison focuses on the contrastive loss, taking the form of either SaNCE, InfoNCE, or no contrastive loss. Accordingly, we select baselines based on contrastive loss: **CCM** [Fu et al., 2021], which utilises InfoNCE, and **TESAC** [Yu et al., 2020], which relies solely on the RL loss, allowing assessment of the context encoder without contrastive loss. Since CCM and TESAC use an RNN encoder, we also include **PEARL** [Rakelly et al., 2019], which utilises an MLP context encoder and follows the similar RL loss. Additionally, Appendix G includes comparisons with DOMINO [Mu et al., 2022] and CaDM [Lee et al., 2020], using the same environmental setup in the MuJoCo benchmark.

Our method.[2] We use Soft Actor-Critic (SAC) [Haarnoja et al., 2018] as the base RL algorithm, training agents for 1.6 million timesteps in each environment (details in Appendix D.3). SaNCE is a simple objective based on MI that can be used to train any context encoder. We integrate SaNCE into two Meta-RL algorithms: (1) **SaTESAC** is TESAC with SaNCE, which uses SaNCE for contrastive learning, using a $|c|$ times smaller sample space (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 involves a robot arm control task using the Franka Emika Panda [Gallouédec et al., 2021], where the robot moves 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 requires high skill diversity; for instance, the agent must use Pick&Place in high-friction tasks and Push in high-mass tasks.

Table 1: Comparison of success rate $\pm$ standard deviation 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 paried 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*

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. The t-test results in Table 1 show that SaMI significantly improves success rates across training, moderate, and extreme test sets at a 0.05 significance level. Video demos[2] show that agents equipped with SaMI acquired multiple skills (Push, Pick&Place) to handle various tasks. When faced with an unknown task, the agents explore by attempting to lift the cube, infer the context, and adjust their skills accordingly within the episode. We visualised the context embeddings using UMAP [McInnes et al., 2020] (Figure 6(a) and Appendix F) and t-SNE [Van der Maaten and Hinton, 2008] (Appendix F), plotting the final step from 100 tests per task. Skills were identified through contact points between the end effector, cube, and table (see Appendix D for more details), and heatmaps [Waskom, 2021] were used to visualise executed skills. Figure 6 shows that SaCCM agents learned the Push skill for large cube masses (30 Kg, 10 Kg) and Pick&Place for smaller masses, while CCM showed no clear skill grouping (Figure 16 in Appendix F.1). Overall, SaMI incentivises agents to autonomously learn diverse skills, enhancing generalisation across a wider range of tasks. Specifically, through the cycle of effective exploration, context inference, and adaptation, diverse skills emerge solely from the data. Further visualisation results are 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

[2]Our code, video demos and experimental data are available at https://github.com/uoe-agents/SaMI

Table 2: Comparison of average return $\pm$ 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***

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 training and test tasks. SaTESAC and SaCCM achieved higher returns in most tasks, except for Ant, Half-Cheetah, and Hopper, where only a single skill was needed. For instance, the Hopper robot learned to hop forward, adapting to different mass values. When environments become complex and require diverse skills for different tasks (Crippled Ant, Crippled Hopper, Crippled Half-Cheetah, SlimHumanoid, HumanoidStandup, and Crippled Walker), SaNCE brings significant improvements. For example, when the Ant robot has 3 or 4 legs available, it learns to roll to generalise across varying mass and damping. In more challenging zero-shot settings, 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 results in Table 2, where SaMI significantly improved performance in extreme test sets. In summary, 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. Our video demos[2] and visualisation results in Appendix F.2 show different skills in all environments.

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

This section analyses whether SaNCE can overcome the $\log$-K curse. During training, environmental features are sampled at the start of each episode, requiring the context encoder to learn the context embedding distribution across multiple tasks. Since InfoNCE samples negative examples from all tasks and SaNCE samples from the current task, SaNCE's negative sample space is $|c|$ times smaller than InfoNCE's. For instance, in the SlimHumanoid environment, where both mass and damping have five values, InfoNCE's sampling space can be 25 times larger than SaNCE's. As shown in Tables 1 and 2, RL algorithms using SaNCE achieve better or comparable performance with significantly fewer negative samples (K) than InfoNCE. This suggests SaNCE effectively addresses the $\log$-K curse, and the SaMI objective helps the contrastive context encoder extract critical information for downstream RL tasks.

The number of negative samples K is influenced by two hyperparameters: ***buffer size***, which determines the negative sample space, and ***contrastive batch size***, which controls the number of samples used to train the contrastive context encoder per update. We analysed these hyperparameters further, and as shown in Figure 7, reductions in buffer and contrastive batch size do not significantly impact the average return for SaCCM and SaTESAC, which maintain state-of-the-art performance with small buffers and batch sizes. The results in Table 2 correspond to a buffer size of 100,000 and a batch size of 12. Results across all environments (refer to Appendix E.2) show that SaNCE exhibits low sensitivity to K, highlighting its potential to overcome the $\log$-K curse.

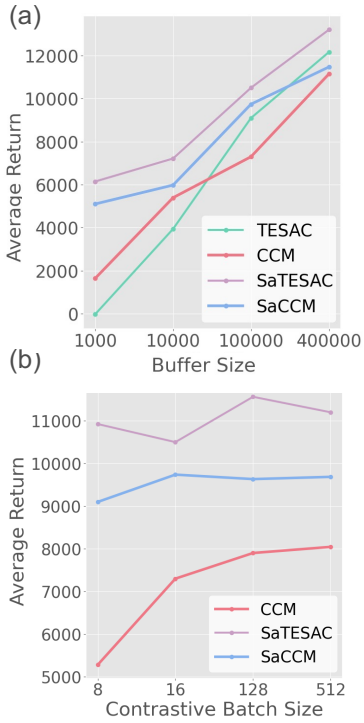


Figure 7: Effect of (a) buffer size (TESAC, CCM, SaTESAC, SaCCM) and (b) contrastive batch size (CCM, SaTESAC, SaCCM) in the SlimHumanoid environment.

6 Conclusion and future work

Zero-shot generalisation has been a longstanding challenge concerning Meta-RL agents, with skill diversity and sample efficiency being key to generalising to previously unseen environments. In this paper, we proposed Skill-aware Mutual Information (SaMI) to learn context embeddings for zero-shot generalisation in downstream RL tasks, and Skill-aware Noise Contrastive Estimation (SaNCE) to optimise SaMI and overcome the $\log$-K curse, along with a practical skill-aware trajectory sampling strategy. Experimental results showed that RL algorithms equipped with SaMI achieved state-of-the-art performance in MuJoCo and Panda-gym benchmarks, particularly in zero-shot generalisation within more complex environments. During the zero-shot generalisation, when faced with an unseen task, SaMI assists agents in exploring effectively, inferring context, and rapidly adapting their skills within the current episode. SaNCE's optimisation uses a significantly smaller negative sample space than baselines, and our analysis on buffer and contrastive batch sizes demonstrated its effectiveness in addressing the $\log$-K curse.

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 [Dunion et al., 2023a]. 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.

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.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76: 201–264, 2023.

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.

Qiang Li. Functional connectivity inference from fmri data using multivariate information measures. *Neural Networks*, 146:85–97, 2022.

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

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. URL `https://arxiv.org/abs/1802.03426`.

Trevor McInroe, Lukas Schäfer, and Stefano V. Albrecht. Multi-horizon representations with hierarchical forward models for reinforcement learning. *Transactions on Machine Learning Research (TMLR)*, 2024.

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.

Leland Gerson Neuberg. Causality: models, reasoning, and inference, by judea pearl, cambridge university press, 2000. *Econometric Theory*, 19(4):675–685, 2003.

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.

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 $Y = \{y_1, \ldots, y_K\}$ of K random samples, containing one positive sample y_1 and $K-1$ negative samples drawn from the distribution $p(y)$, a K-sample InfoNCE estimator is obtained by comparing pairs sampled from the joint distribution $(x, y_1) \sim p(x, y)$ with pairs (x, y_k), constructed using the set of negative examples $y_{2:K}$. InfoNCE compares the positive pairs (x, y_1) with the 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 that 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 $I(x; y) \geq I_{\text{InfoNCE}}(x; y|\psi, K)$ according to:

Proposition 1 *[Poole et al., 2019] A K-sample estimator is an asymptotically tight lower bound on 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]. With a sufficiently powerful deep learning model for ψ and a finite sample size K, such that $I(x; y) \geq \log K$, we can reasonably expect that $I_{\text{InfoNCE}} \approx \log K$ 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 the context encoder ψ influences 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 that preserves the MI between x and y, where $\propto$ stands for 'proportional to' (i.e., up to a multiplicative constant). Substituting

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

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

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:

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

Moreover, according to Equation (7), as the sample size $K \to +\infty$, the K-sample InfoNCE bound becomes sharp and approaches 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 that SaNCE is a K-sample SaNCE estimator and is upper bounded by $\log K$. Since $\frac{f_\psi(c,\pi_c,\tau_c^+)}{f_\psi(c,\pi_c,\tau_c^+)+\sum_{k=2}^K f_\psi(c,\pi_c,\tau_{c,k}^-)} \leq 1$ [Mu et al., 2022], we have:

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

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

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

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

The required sample size is $K^*_{\text{SaNCE}} = |c_1| \cdot |\pi| \cdot M = |\pi| \cdot M$. As $K \to K^*_{\text{SaNCE}}$, we have $I_{\text{SaNCE}}(c; \pi_c; \tau_c|\psi, K) \approx I_{\text{SaMI}}(c; \pi_c; \tau_c)$. 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, while 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 increases significantly due to multiple environmental features $e = \{e^0, e^1, \ldots, e^N\}$. The number of different tasks $|c|$ grows exponentially due to the permutations and combinations of the N environmental features. When the current task e_1 has context embedding c_1, the $c_{2:|c|}$ refer to the context embeddings for the other tasks. As $K \to K^*_{\text{InfoNCE}}$, we have $I_{\text{InfoNCE}}(c; \pi_c; \tau_c|\psi, K) \approx I_{\text{SaMI}}(c; \pi_c; \tau_c)$. Thus, 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. According to the definition of SaMI in Equation (2), we have $I_{\text{SaMI}}(c; \pi_c; \tau_c) \leq I(c; \tau_c)$, as illustrated using Venn diagrams in Figure 8 (a) and (b). SaMI is formulated based on interaction information [McGill, 1954], which aims to capture the relationships among multivariate variables by quantifying the amount of information (redundancy or synergy) shared among three variables. Interaction information has been less extensively studied, partly due to its challenging interpretation from both information theory and neuroscience perspectives, as it can be either positive or negative [Li, 2022].

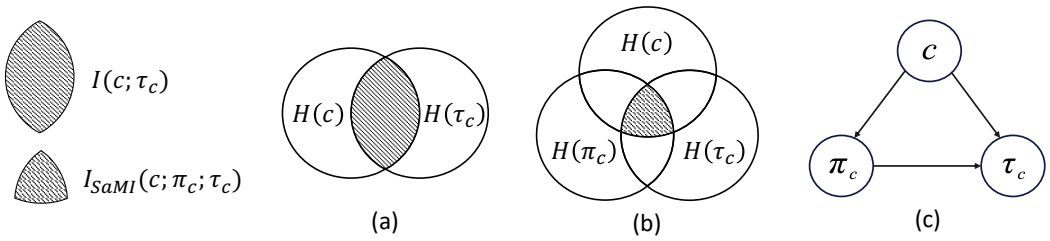


Figure 8: Venn diagrams illustrating (a) mutual information $I(c; \tau_c)$, (b) interaction information $I_{\text{SaMI}}(c; \pi_c; \tau_c)$, and (c) the MDP graph of the context embedding c, skill π_c, and trajectory τ_c, which represents a common-cause structure [Neuberg, 2003].

In the Meta-RL setting, the MDP (causal) graph structure, shown in Figure 8 (c), illustrates that the context embedding c, skill π_c, and trajectory τ_c form a common-cause structure, where c acts as a shared cause influencing both the skill π_c and the trajectory τ_c. Consequently, $I(\pi_c, \tau_c \mid c) < I(\pi_c, \tau_c)$. Therefore, $I_{\text{SaMI}}(c; \pi_c; \tau_c) > 0$ is guaranteed, making it interpretable in real-world robotic control tasks.

Putting it together. Thus, we establish Lemma 2: we always have $I_{\text{InfoNCE}}(c; \pi_c; \tau_c|\psi, K) \leq I_{\text{SaNCE}}(c; \pi_c; \tau_c|\psi, K) \leq \log K \leq I_{\text{SaMI}}(c; \pi_c; \tau_c) \leq I(c; \tau_c)$, as shown in Figure 3, while learning a skill-aware context encoder ψ with the SaNCE estimator. Since $K^*_{\text{SaNCE}} \ll K^*_{\text{InfoNCE}}$, $I_{\text{SaNCE}}(c; \pi_c; \tau_c|\psi, K)$ serves as a much tighter lower bound for 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 $I_{\text{Sa+InfoNCE}}$

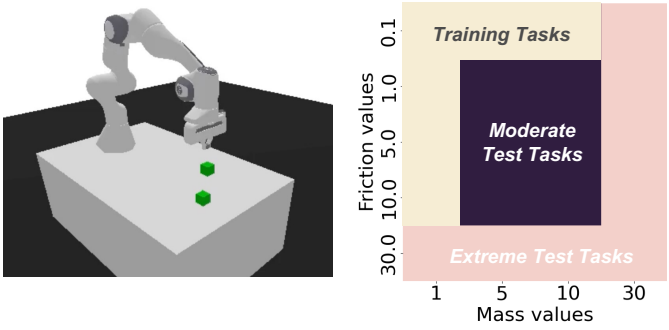
In this section, we illustrate the sample size of $I_{\text{Sa+InfoNCE}}(c; \pi_c; \tau_c|\psi, K)$. Sa+InfoNCE incorporates SaNCE into InfoNCE, using positive samples $\tau^+_{c_1}$ from task e_1 after executing skill $\pi^+_{c_1}$, and negative

samples are trajectories $\tau^-_{c_{1:K}}$ from executing skills $\pi^-_{c_{1:K}}$ in tasks $e_{1:K}$, respectively. Therefore, this approach is equivalent to first observing the variable c and then observing the variable π_c, i.e., sampling from the distribution $p(\pi_c, \tau_c|c)p(c)$. We have:

$$
\begin{aligned}
&I_{\text{Sa+InfoNCE}}(c; \pi_c; \tau_c|\psi, K) \\
&= \mathbb{E}_{p(c)p(\pi^+_{c_1}, \tau^+_{c_1}|c)p\left(\left(\pi^-_{2:|c|}, \tau^-_{2:|c|}\right)_{2:K}\right)} \left[\log\left(\frac{K \cdot f_\psi(c, \pi^+_{c_1}, \tau^+_{c_1})}{f_\psi(c, \pi^+_{c_1}, \tau^+_{c_1}) + \sum_{k=2}^{K} f_\psi(c, \pi^-_{2:|c|,k}, \tau^-_{2:|c|,k})}\right)\right] \\
&\leq \mathbb{E}_{p(c)p(\pi^+_{c_1}, \tau^+_{c_1}|c)p\left(\left(\pi^-_{2:|c|}, \tau^-_{2:|c|}\right)_{2:K^*_{\text{Sa+InfoNCE}}}\right)} \left[\log\left(\frac{K^*_{\text{Sa+InfoNCE}} \cdot f_\psi(c, \pi^+_{c_1}, \tau^+_{c_1})}{f_\psi(c, \pi^+_{c_1}, \tau^+_{c_1}) + \sum_{k=2}^{K^*_{\text{Sa+InfoNCE}}} f_\psi(c, \pi^-_{2:|c|,k}, \tau^-_{2:|c|,k})}\right)\right] \\
&\approx \log K^*_{\text{Sa+InfoNCE}} \qquad (12)
\end{aligned}
$$

It should be noted that such a combination increases the size of the negative sample space, i.e., $K^*_{\text{Sa+InfoNCE}} = \left(\sum_{i=1}^{|c|} |\pi^-_{c_i}| + |\pi^+_{c_1}|\right) \cdot M \geq K^*_{\text{SaNCE}}$. The misaligned bars in Figure 4 (c) illustrate that negative sample spaces may vary across tasks. This variation arises because we define negative samples as trajectories with low returns, making the size of the negative sample space influenced by sampling randomness. With the same number K of samples, $I_{\text{Sa+InfoNCE}}$ is less precise and looser than I_{SaNCE}. Therefore, 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 9: (a) Modified Panda-gym benchmarks, (b) the training tasks, moderate test tasks, and extreme test tasks. The moderate test task setting involves combinatorial interpolation, while the extreme test task setting includes unseen ranges of environmental features and represents 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 to 0 [3] and maintaining the freedom of the grippers [4], allowing the agent to explore whether it should push or grasp the cube. ***Skills*** in this benchmark are defined as:

- *Pick&Place skill*: This skill specifically refers to the agent using the gripper to grasp the cube, lift it 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.

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

[4] In the original Push task, the grippers are blocked to ensure the agent can only push cubes. However, this restriction prevents the agent from learning Pick&Place skills, leading to "unpushable" failure in Figure 1 (b).

- *Push skill*: This skill involves the agent moving the cube on the table to the goal position, either by dragging or sliding it. We confirm the Push skill by detecting that the cube's height equals half its width.

- *Other skills*: Any behaviour modes other than Pick&Place and Push are classified as other skills.

Some elements in the RL framework are defined as follows:

State space: We use feature vectors that contain the cube's 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 states.

Action space: The action space has 4 dimensions; the first three dimensions represent changes in the end-effector's position, and the last dimension represents the change in the gripper's width.

During training, we randomly select a combination of environmental features from a training set by sampling combinations from the following 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 on all tasks from the moderate test setting, where mass $\in \{5.0, 10.0\}$ and friction $\in \{1.0, 5.0, 10.0\}$ (shown in Figure 9(b)), and on all tasks from the extreme test 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 9(b)).

D.2 Modified MuJoCo

We extended the modified MuJoCo benchmark introduced in DOMINO [Mu et al., 2022] and CaDM [Lee et al., 2020]. In our extension, four new environments were added (Walker, Crippled Hopper, Crippled Walker, HumanoidStandup), compared to the original benchmark. Additionally, in our experiments, we used a different task set design (Table 3) than those used in the DOMINO and CaDM papers. For Hopper, Walker, Half-Cheetah, Ant, HumanoidStandup, and SlimHumanoid, we used the MuJoCo physics engine environments and implemented settings from Clavera et al. [2019b] and Seo et al. [2020], scaling the mass of every rigid link by a scale factor m and the damping of every joint by a scale factor d. For Crippled Ant, Crippled Hopper, Crippled Walker, and Crippled Half-Cheetah, we used the implementation available from Seo et al. [2020], scaled the mass of each rigid link by a factor of m, scaled the damping of each joint by a factor of d, and randomly selected joints to be uncontrollable (i.e., masking the corresponding actions with 0). Generalisation performance is measured in two different regimes: moderate and extreme, where the moderate regime draws environmental features from a range closer to the training range compared to the extreme regime. Our settings for training, extreme, and moderate test tasks are provided in Table 3.

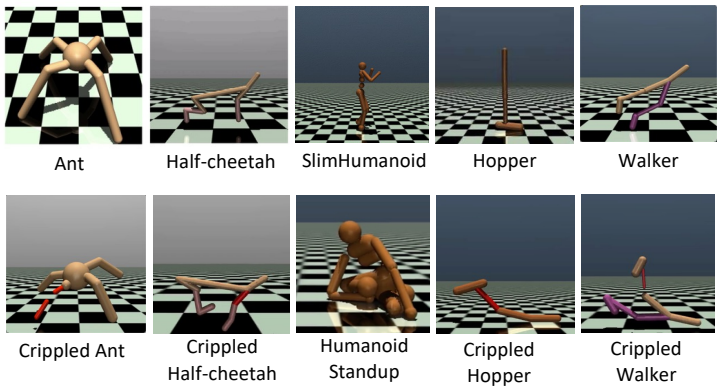


Figure 10: Ten environments in modified MuJoCo benchmark.

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 $R_1 = \varnothing$ or $R_1 \in \{0, 1, 2, 3\}$	$m \in \{0.40, 0.50, 1.50, 1.60\}$ $d \in \{0.40, 0.50, 1.50, 1.60\}$ Crippled Legs $R_1 \in \{0, 1, 2, 3\}$	$m \in \{0.20, 0.40, 1.60, 1.80\}$ $d \in \{0.20, 0.40, 1.60, 1.80\}$ Crippled Legs $\{R_1, R_2\} \in \{0, 1, 2, 3, 4, 5\}$ $(R_1 \neq R_2)$	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 the Panda-gym and MuJoCo benchmarks. Most hyperparameter values remain unchanged across tasks, except for the contrastive batch size and the 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

D.3 Implementation details

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

Base algorithm. We use SAC [Haarnoja et al., 2018] for the downstream evaluation of the learned context embedding. SAC is an off-policy actor-critic method that leverages the maximum entropy framework for 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. During 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 evaluated 100 episodes in each environment, with tasks randomly sampled from the moderate and extreme task sets.

Encoder architecture. In our method, the context encoder ψ is modelled as a Long Short-Term Memory (LSTM) network that produces a 128-dimensional hidden state vector, which is subsequently processed through a single-layer feed-forward network to generate a 6-dimensional context embedding. We aim for the agent to complete the three steps of "explore effectively, infer, adapt" within an episode. Therefore, we initialise the hidden state and cell state of the LSTM to zero at the start of each episode. Both the actor and critic 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^*} as follows:

$$\theta_{\psi^*} \leftarrow m \cdot \theta_\psi + (1 - m) \cdot \theta_{\psi^*}. \tag{13}$$

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 (13) 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 optimised 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 with equal weight, setting $\alpha = 1.0$ for the MuJoCo benchmark and $\alpha = 0.01$ for the Panda-gym benchmark. Additionally, we analyse the effect of the loss coefficient α for CCM, SaTESAC, and SaCCM in the MuJoCo (Figure 12) and Panda-gym (Figure 11) benchmarks.

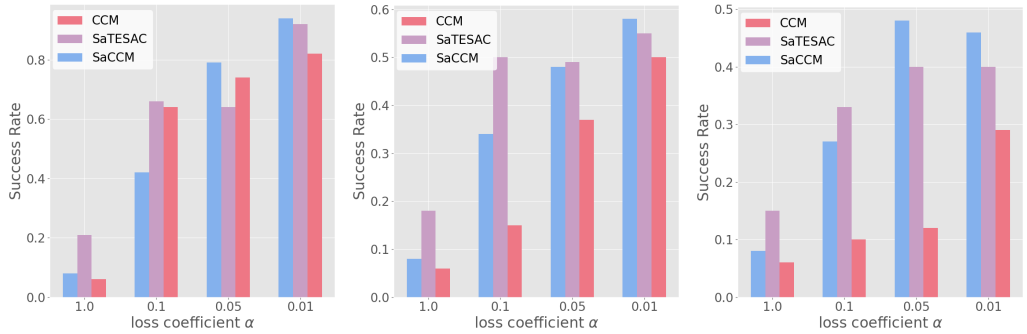


Figure 11: Loss coefficient α analysis of Panda-gym benchmark in training and test (moderate and extreme) tasks.

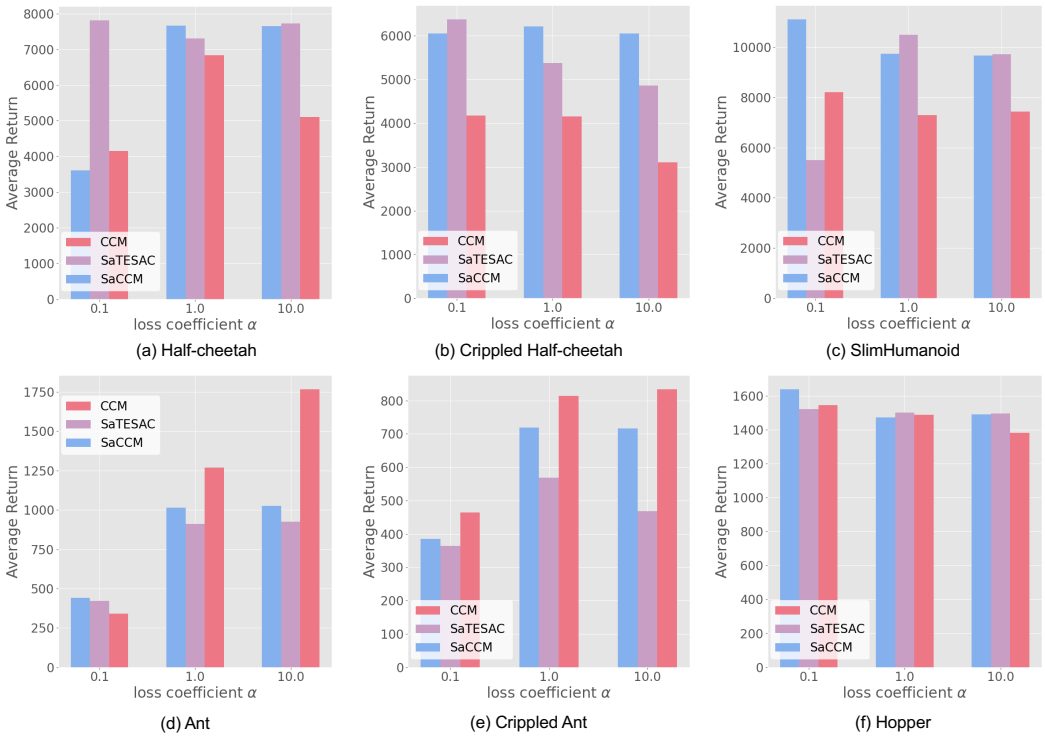


Figure 12: Loss coefficient α analysis of MuJoCo benchmark in training tasks.

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

E.2.1 Buffer size

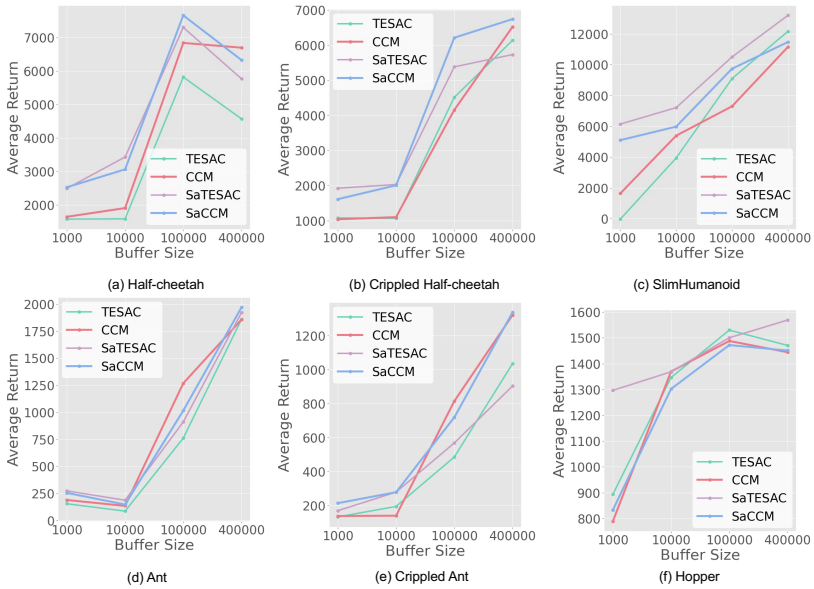


Figure 13: Comparison of different buffer sizes in the MuJoCo benchmark on training tasks (averaged over 5 seeds). Buffer sizes are 400,000, 100,000, 10,000, and 1,000.

E.2.2 Contrastive batch size

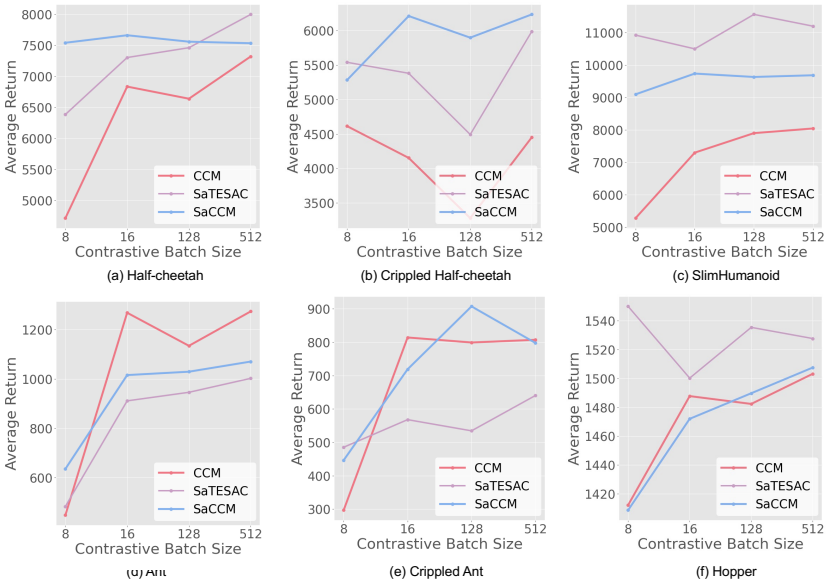


Figure 14: Comparison of different contrastive batch sizes in the MuJoCo benchmark on training tasks (averaged over 5 seeds). Contrastive batch sizes are 512, 128, 16, and 8.

F Further skill analysis

F.1 Panda-gym

F.1.1 Visualisation of context embedding

We visualise the context embedding using UMAP [McInnes et al., 2020] (Figure **??**) and t-SNE [Van der Maaten and Hinton, 2008] (Figure 16). When the mass of the cube is high (30 kg and 10 kg), the agent learned the Push skill (indicated by the yellow bounding box in Figure 1(a)), whereas with lower masses, the agent learned the Pick&Place skill. However, as shown in Figure 16(b), CCM did not display clear skill grouping. This indicates that SaMI extracts high-quality skill-related information from the trajectories and enables agents to autonomously discover a diverse range of skills for handling multiple tasks.

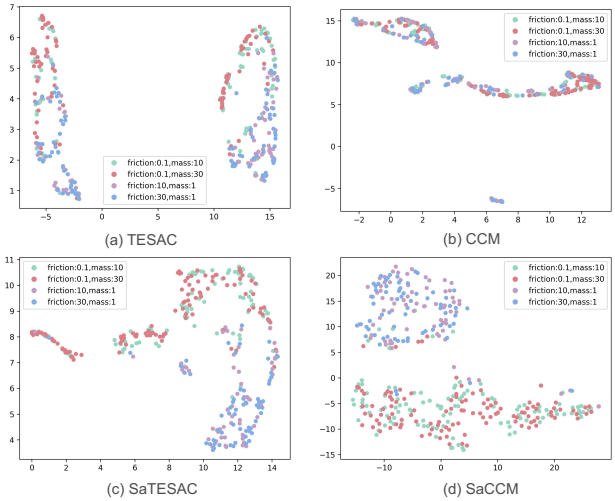


Figure 15: UMAP visualisation of context embeddings extracted from trajectories collected in the Panda-gym environments.

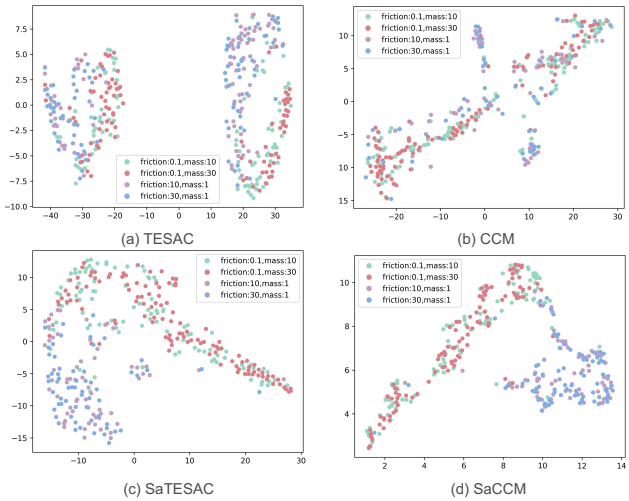


Figure 16: t-SNE visualisation of context embeddings extracted from trajectories collected in the Panda-gym environments.

F.1.2 Heatmap of Panda-gym benchmark

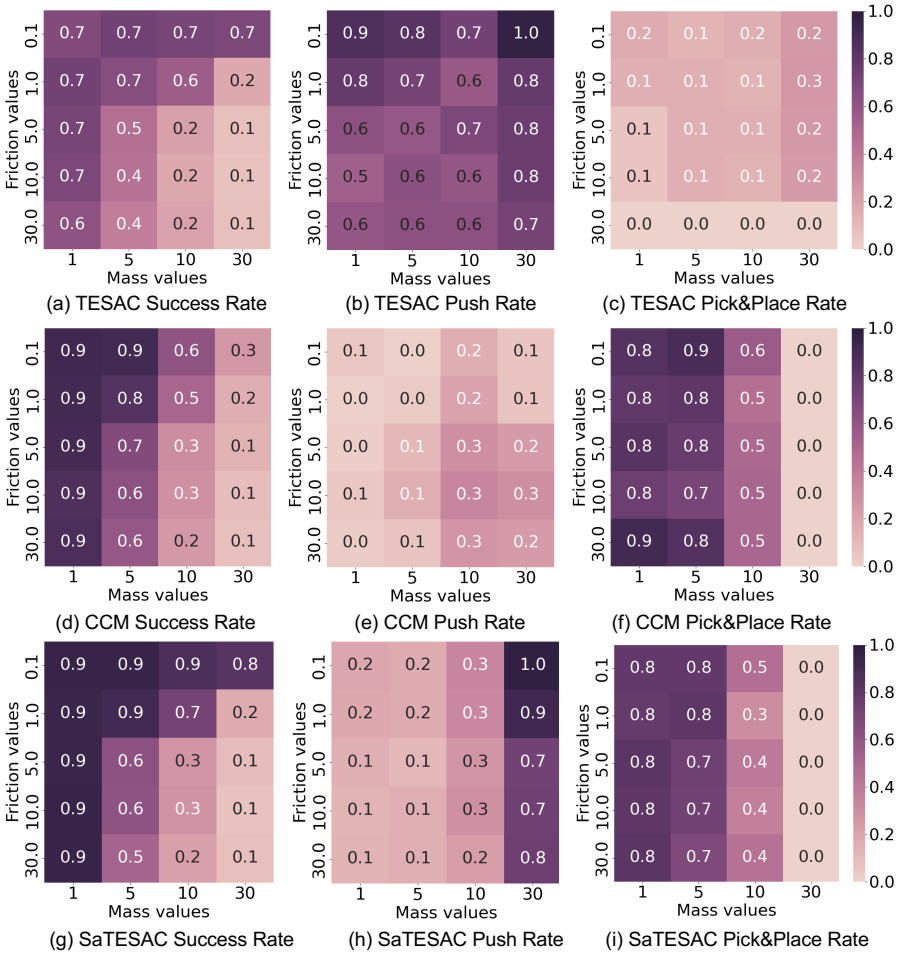


Figure 17: Heatmap of success rates and learned skills of SaCCM. For each grid, the $(i, j)^{\text{th}}$ location shows the probability of the skills executed over 100 evaluations with $(\text{mass} = i, \text{friction} = j)$.

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

In contrast, as shown in Figures 17(e-f), CCM predominantly learned the Pick&Place skill, resulting in a drop in success rates for tasks with $mass = 30$, as the agent could not lift the cube off the table using the Pick&Place skill, as depicted in Figure 17(d). The visualisation of the context embedding (Figure 16) did not reveal clear skill grouping across different tasks.

Finally, TESAC primarily 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 manipulate its fingers to pick up cubes. Consequently, TESAC's success rate notably decreased in environments with higher friction.

F.2 MuJoCo

SaMI enables RL agents to be versatile and embody multiple skills. Additionally, video demos (available at https://github.com/uoe-agents/SaMI) provide a better demonstration of the different skills.

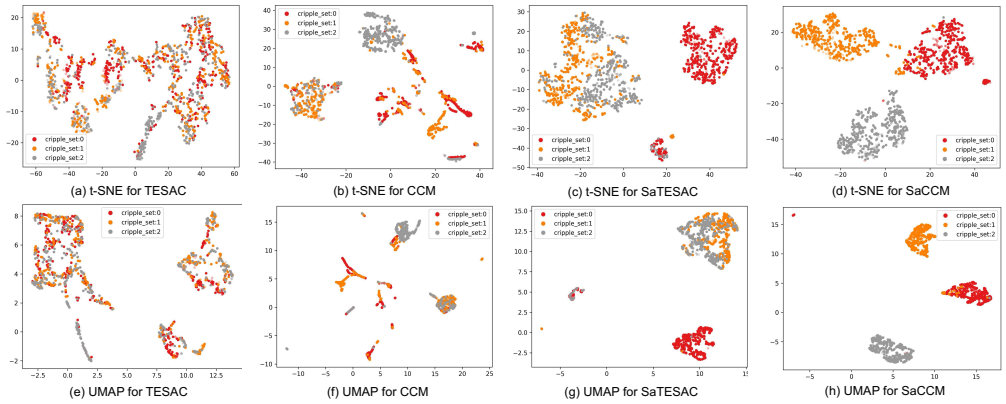


Figure 18: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in the Crippled Half-Cheetah environment. "cripple_set" refers to the index of the crippled joint. The figure shows the context embeddings of tasks in three moderate test settings. Combined with the video demos[2] for skill analysis, the Crippled Half-Cheetah robot executed three distinct forward running skills.

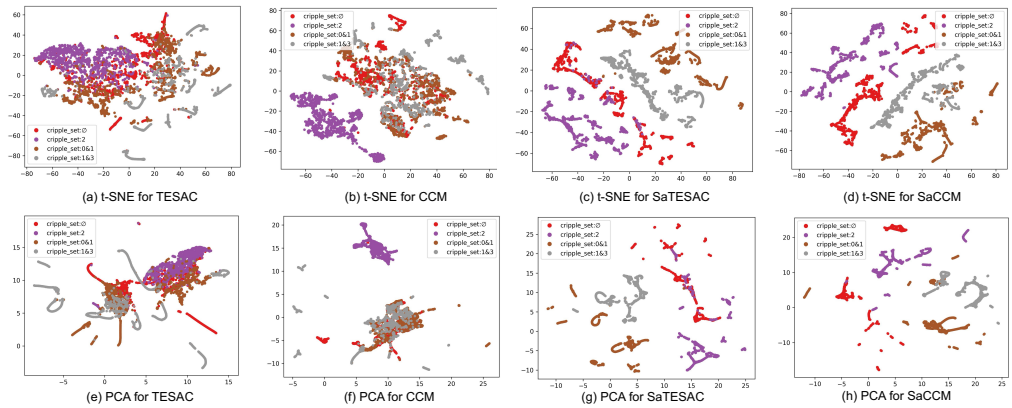


Figure 19: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in the Crippled Ant environment. "cripple_set" refers to the index of the crippled joint. When 3 or 4 legs are available, the Ant robot (trained with SaCCM) rolls to adapt to varying mass and damping. However, with only 2 adjacent legs during zero-shot generalisation, it switches to walking. If 2 opposite legs are available, the Ant can still roll but eventually tips over. Please refer to the video demos[2].

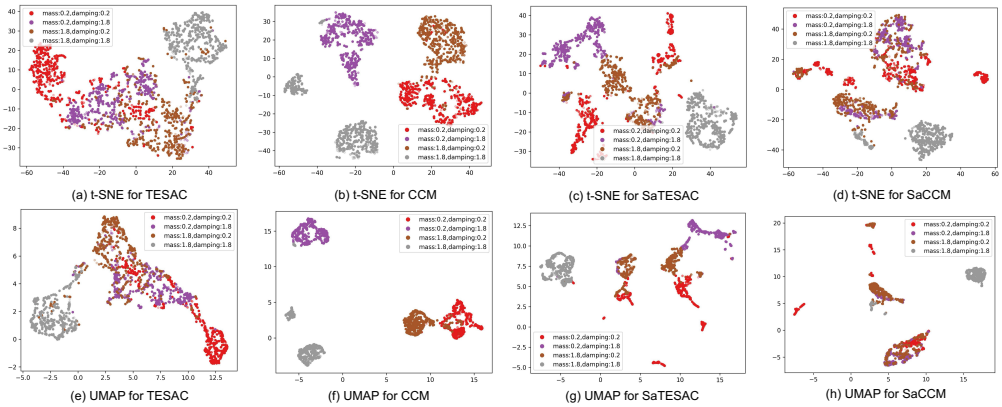


Figure 20: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in the Half-Cheetah environment. During zero-shot generalisation, SaCCM demonstrates different skills for running forwards at various speeds, as well as skills for doing flips and faceplanting.

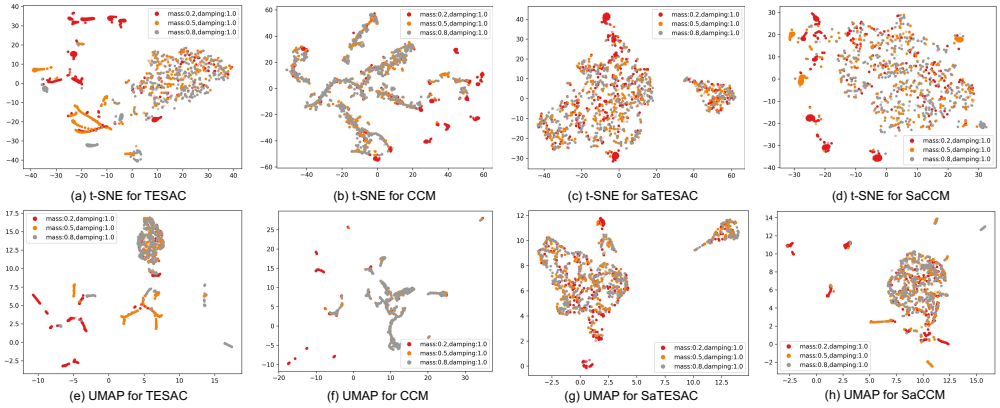


Figure 21: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in Ant environment. The Ant robot learned a single skill, rolling, to adapt to different mass and damping values.

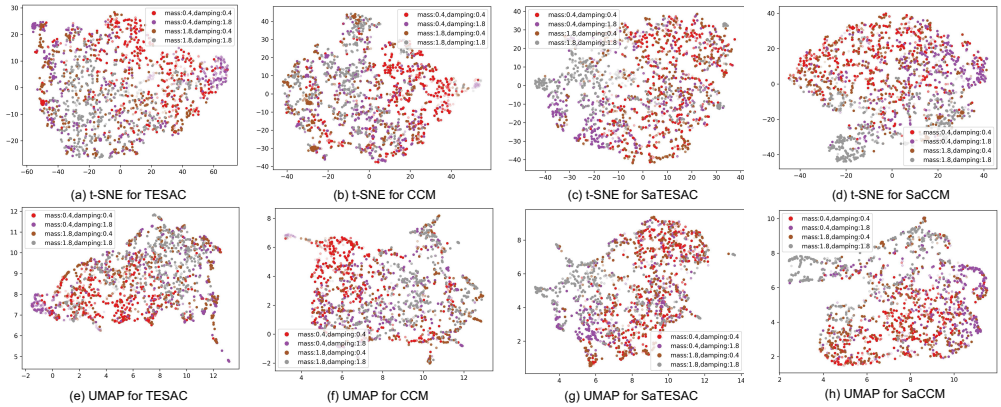


Figure 22: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in SlimHumanoid environment. The Humanoid Robot crawls on the ground using one elbow. When the damping is relatively high (damping=1.8), the Humanoid Robot can crawl forward stably, but when the damping is low (damping=0.4), it tends to roll.

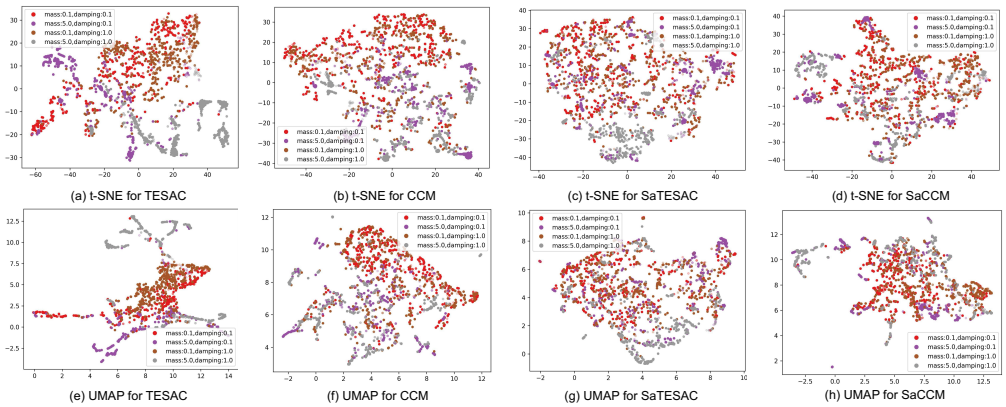


Figure 23: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in HumanoidStandup environment. SaCCM and SaTESAC learned a sitting posture that makes it easier to stand up, allowing it to generalise well when mass and damping change.

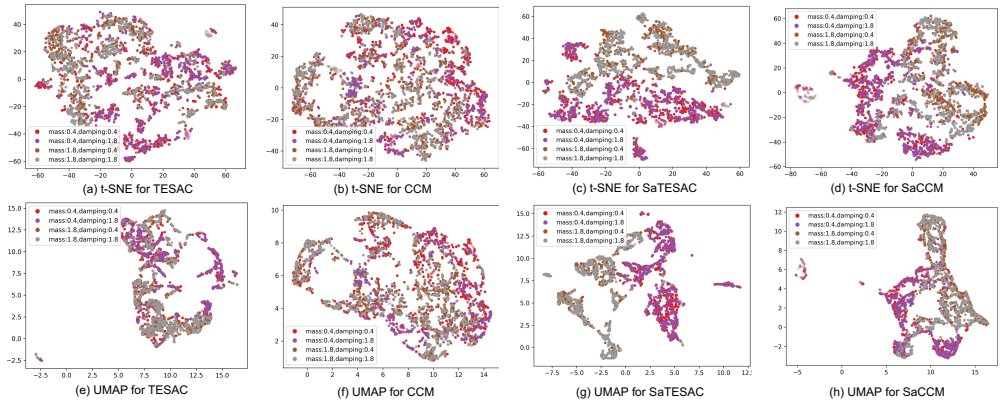


Figure 24: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in Hopper environment. Combined with the video demos [2] for skill analysis, the plots for SaCCM and SaTESAC show two skills: 1) when the mass is low, the Hopper hops in an upright posture; 2) when the mass is higher, the Hopper hops forward on the floor.

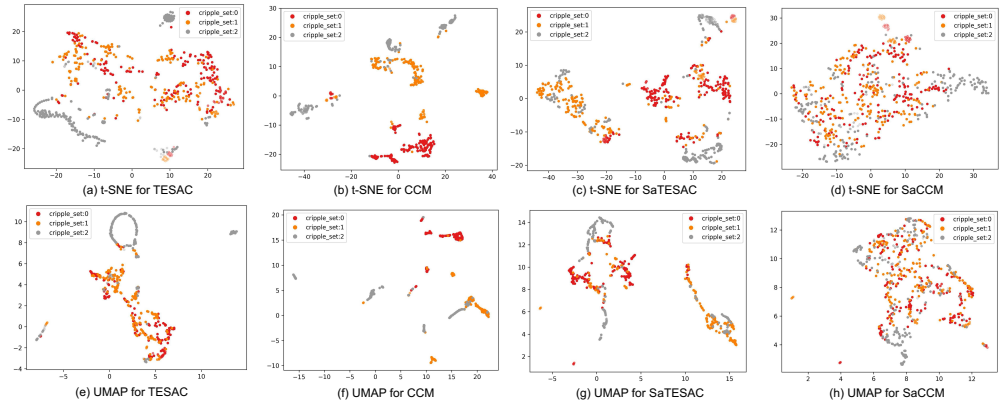


Figure 25: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in Crippled Hopper environment. "cripple_set" refers to the index of the crippled joint. SaCCM and SaTESAC learned to take a big hop forward at the beginning (i.e., effective exploration) and then switch to different skills based on environmental feedback.

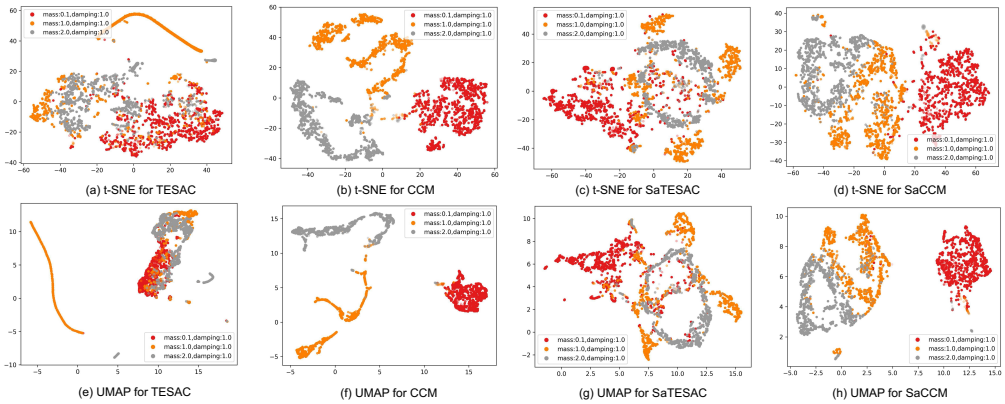


Figure 26: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in Walker environment. The Walker learned a single skill, hopping forward on the floor through both right and left legs.

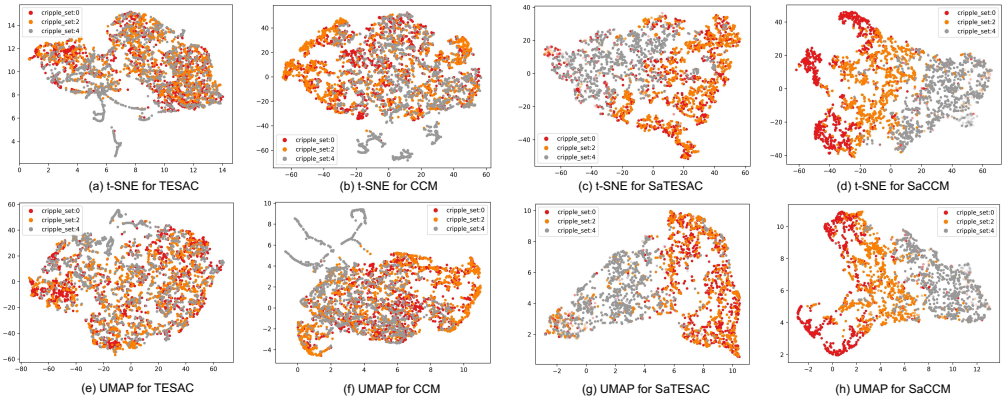


Figure 27: t-SNE and UMAP visualisations of context embeddings extracted from trajectories collected in Crippled Walker environment. "cripple_set" refers to the index of the crippled joint. The Crippled Walker (trained with SaTESAC and SaCCM) learned to hop forward using the right leg (cripple_set:0 and cripple_set:2) in the training and moderate test tasks and switched to hopping forward using the left leg (cripple_set:4) in the extreme test tasks.

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
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 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 MI 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 (paried 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 (paried 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	0.000260*	0.000160*	0.001310*
SaCCM-CCM	0.000230*	0.002190*	0.001390*

H Statistical hypothesis tests (paried 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 UMAP 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.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract and introduction clearly state the claims made, including the contributions made in the paper and important assumptions and limitations.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalise to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations and future work of this work in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Proofs are provided in Appendix A and B.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The code will be open-sourced and all the implementation details for both the environment setup and algorithms are in the Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will open-source our modified benchmarks (i.e., data) and code after the double-blind review phase ends.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have provided experimental details in Appendix D including hyperparameters, how they were chosen, type of optimizer, etc.

   Guidelines:but

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the mean and standard deviation of our environmental results over five seeds.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have provided sufficient information on the computer resources (type of compute workers, memory, time of execution) in Appendix D.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We have fully considered the potential societal impacts of our research. Our work is conducted in simulated environments, and there is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets used in our study are open-source, and we have acknowledged the authors' copyrights in the References.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide detailed descriptions of the assets we will release in Appendix D and on our open-source homepage. During the double-blind review phase, we have anonymised the URLs.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.