Learning to Undo: Transfer Reinforcement Learning under State Space Transformations

Mridul Mahajan[1] **Aldo Pacchiano**[1,2†] **Xuezhou Zhang**[1†]
[1]Boston University
[2]Broad Institute of MIT and Harvard
{mridulm,pacchian,xuezhouz}@bu.edu

Abstract

Transfer learning in reinforcement learning (RL) has shown strong empirical success. In this work, we take a more principled perspective by studying when and how transferring knowledge between MDPs (a source and a target) can be provably beneficial. Specifically, we consider the case where there exists an undo map such that applying this map to the target's state space recovers the source exactly. We propose an algorithm that learns this map by matching state feature statistics gathered from both MDPs, and then uses it to transfer the source policy to the target. We theoretically justify the algorithm by analyzing the setting when the undo map is linear and the source is linearly-$Q^\star$ realizable, where our approach has strictly better sample complexity than tabula rasa RL in the target MDP. Empirically, we demonstrate that these benefits extend beyond this regime: on challenging continuous control tasks and Atari games, our method achieves significantly better sample efficiency. Overall, our results highlight how shared structure between tasks can be leveraged for efficient transfer of policies across environments. Our codebase is publicly available at `github.com/MridulMahajan44/neurips2026-learning-to-undo`.

1 Introduction

Reinforcement learning (RL) has seen rapid progress in recent years and has achieved strong performance in complex tasks, such as video games, locomotion, manipulation, and navigation [Mnih et al., 2015, Lee et al., 2019, Zhu et al., 2019]. Despite this, learning high-quality policies from scratch typically requires millions of environment interactions, which limits the applicability of RL in real-world domains where data collection is costly or time-consuming [Dulac-Arnold et al., 2019].

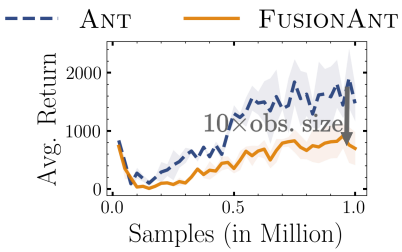
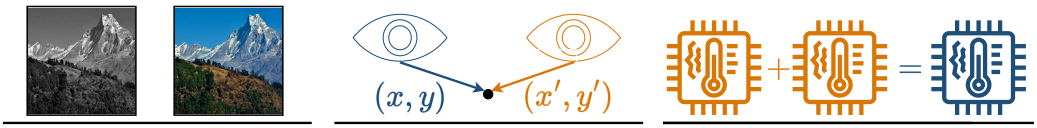


Figure 1: State space transformations can significantly decrease sample efficiency (e.g., increasing observation size by $10\times$).

Transfer learning aims to overcome this challenge by leveraging knowledge learned from source tasks to accelerate learning on related target tasks [Taylor and Stone, 2009]. It has the potential to drastically improve sample efficiency by effectively reusing prior experience. While transfer learning has seen strong empirical success in RL [Zhu et al., 2020], our goal is not to design a state-of-the-art transfer method that works across arbitrary variations in tasks. Instead, we study the problem from a more principled lens, where we explicitly model the similarity between tasks and exploit it to obtain an algorithm that not only has provable sample complexity improvements over learning from scratch, but also works empirically.

[†]Equal senior authorship.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



Color Transformation | Different Frame of Reference | Sensor Fusion

Figure 2: State space transformations are grounded in real-world applications.

An interesting and practically relevant setting with such structure is that of state space transformations: the source and target tasks differ due to transformations in the state space. For instance, real-world transformations such as color transformation from grayscale to RGB, change in frame of reference, and sensor fusion – all are state space transformations (see Fig. 2). Such transformations provide a rich and practical setting to design principled transfer learning methods that explicitly model them. One might assume that such transformations are trivial to handle since the underlying task remains the same. However, this intuition is misleading: as shown in Fig. 1, even a $10\times$ increase in observation size in the sensor fusion setting, where readings must be aggregated meaningfully, is enough to render tabula rasa RL severely suboptimal, which highlights that principled transfer is necessary.

State-space and observation transformations naturally occur in many practical RL scenarios where the underlying task dynamics remain unchanged, but the agent's interface to the environment changes. For instance, in robotics, policies are often trained in simulation and then deployed on real systems, since real-world training is significantly more costly and difficult to parallelize [Zhao et al., 2020]. In the transition from simulation to the real system, differences between simulated and physical sensors cause discrepancies in sensing and perception, and thus, introduce shifts in the agent's observation space [Bohez et al., 2017]. In such settings, relearning policies from scratch can be very sample-inefficient even though the underlying control problem essentially remains the same.

Furthermore, low-dimensional privileged state features are often available during development, but deployment must operate under high-dimensional inputs such as camera observations. Even when both types of observations are available, learning with compact representations is often substantially easier [Hu et al., 2024], which highlights the utility of learning under state-space transformations.

In this work, we address this setting by learning an *undo map* that transforms the state space in the target back to the source. Instead of learning a policy from scratch, we recover a function that *undoes* the transformation applied to the target states, which allows reusing source policies in the target task.

To this end, we introduce a distributional notion of *undo discrepancy* that measures the quality of a candidate undo map. Minimizing this discrepancy is equivalent to matching the state feature distributions in the source MDP and the corresponding *undone* target feature distributions at each timestep under open-loop action sequences. We then propose a practical transfer learning algorithm that optimizes this objective using samples from the target. Theoretically, we show that when the undo map is linear and the source is linearly-$Q^\star$ realizable, our approach has strictly better sample complexity than learning from scratch.

To empirically evaluate our method, we use challenging continuous control tasks and Atari games, and construct target tasks by applying transformations such as changes in frame of reference and coordinate system, sensor fusion-inspired transformations, and color shifts. We also consider perception gaps such as moving from sensor observations in the source to pixel observations in the target, and moving from semantic bitmaps in the source to pixel observations in the target. Our results show that learning the undo map consistently outperforms learning the target policy from scratch in terms of sample efficiency, where it often achieves near-optimal performance with an order of magnitude fewer samples. These results highlight the practical utility of our approach for transfer RL.

To summarize, our work makes the following contributions:

I. We propose a novel transfer learning method, LUMOS (**L**earning to **U**ndo by **M**atching **O**bservation **S**tatistics), that performs regression on state feature statistics to *undo* state-space transformations in the target MDP (Sections 3.1 and 3.2).

II. Theoretically, we prove that our method achieves strictly better sample complexity than learning from scratch when the undo map is linear and the source is linearly-$Q^\star$ realizable (Section 3.3).

III. We empirically demonstrate the practicality of our method via experiments on challenging continuous control tasks and Atari games involving state-space transformations (Section 4).

2 Problem Setup

Tasks. We model each task as a finite-horizon episodic Markov Decision Process (MDP), defined by the 7-tuple $(\mathcal{S}, \mathcal{A}, \mathbb{P}, r, H, \mu, \phi)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ defines the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, H is the fixed horizon length, $\mu \in \mathcal{P}(\mathcal{S})$ is the initial state distribution, and $\phi : \mathcal{S} \to \mathbb{R}^d$ is a fixed feature map. W.l.o.g., we assume that $d \leq H$. The agent interacts with the task using a non-stationary policy $\pi_h : \mathbb{R}^d \to \mathcal{P}(\mathcal{A})$, and for $0 \leq h < H$, chooses an action $a_h \sim \pi_h(\cdot \mid \phi(s_h))$. The value of the policy π is defined as the expected return $V^\pi \triangleq \mathbb{E}_{s_0 \sim \mu, \pi, \mathbb{P}} \left[\sum_{h=0}^{H-1} r(s_h, a_h)\right]$, and the agent's goal is to learn a policy $\pi^\star$ that maximizes this quantity, i.e., $V^* = \max_\pi V^\pi$. The state-action value function $Q^\pi(s, a)$ denotes the expected return when taking action a in state s, and following policy π thereafter.

Transfer Learning. The transfer learning setting follows the protocol in which the agent interacts with a source task $\mathcal{M}_S = (\mathcal{S}^S, \mathcal{A}^S, \mathbb{P}^S, r^S, H, \mu^S, \phi_S)$ to learn a policy π_S, and then leverages the learned policy to speed up learning in the target task $\mathcal{M}_T = (\mathcal{S}^T, \mathcal{A}^T, \mathbb{P}^T, r^T, H, \mu^T, \phi_T)$. We assume that the agent has complete access to the source MDP $\mathcal{M}_S$, either by knowing its dynamics and reward function exactly or by being allowed to interact with it arbitrarily without incurring any sample complexity cost. In contrast, interactions with the target MDP $\mathcal{M}_T$ are costly. Further, the target MDP does not provide reward feedback during adaptation.

Shared Structure and Undo Map. We assume that both tasks are exactly the same except that the agent observes states through task-specific feature maps: $\phi_S : \mathcal{S} \to \mathbb{R}^{d_S}$ in the source and $\phi_T : \mathcal{S} \to \mathbb{R}^{d_T}$ in the target. These feature maps are related by an unknown transformation (the *undo* map) $U_\star : \mathbb{R}^{d_T} \to \mathbb{R}^{d_S}$, such that for all $s \in \mathcal{S}$, $\phi_S(s) = U_\star\big(\phi_T(s)\big)$. This relationship induces a correspondence between policies: for any policy $\pi_S : \mathbb{R}^{d_S} \to \mathcal{P}(\mathcal{A})$ defined in the source feature space, we define the corresponding policy in the target task as $\pi_T\big(\phi_T(s)\big) = \pi_S\Big(U_\star\big(\phi_T(s)\big)\Big)$. These policies are equivalent in terms of their value functions: $V_S^{\pi_S}(s) = V_T^{\pi_T}(s)$ for all $s \in \mathcal{S}$. This allows source policies to transfer to the target despite differing state representations.

Objective. Given complete access to the source MDP $\mathcal{M}_S$, our objective is to learn a near-optimal policy for the target MDP $\mathcal{M}_T$ using as few samples from the target as possible.

3 LUMOS: Learning to Undo by Matching Observation Statistics

In this section, we show how the undo map can be learned by leveraging the invariance of the action space across the source and target MDPs. First, we formally define a distributional measure to evaluate the quality of a candidate undo map (Section 3.1). We then propose a practical learning algorithm, LUMOS (Section 3.2), that matches observation statistics between the source MDP and the target MDP viewed through an undo map. This algorithm, illustrated in Fig. 3, enables transfer of the source policy by composing it with the learned undo map. Theoretically, under certain assumptions, we show that LUMOS enjoys strictly better sample complexity than learning from scratch (Section 3.3).

3.1 Distributional Measure of Undo Map Quality

To estimate the undo map, we match state feature distributions induced by an *open-loop* sequence of actions in the source MDP $\mathcal{M}_S$ with those induced by the *undone* trajectories in the target MDP $\mathcal{M}_T$. This leverages the fact that while the observations differ, the underlying dynamics remain consistent.

For an arbitrary undo map U, we now introduce the notion of source and *undone* target state features.

Definition 3.1 (Source and Undone Target Features)**.** Given an open-loop action sequence $a_{0:h-1}$, define the source feature random variable at timestep h as

$$\psi_h^S(a_{0:h-1}) \triangleq \phi_S(s_h), \tag{1}$$

and, for an arbitrary undo map U, define the *undone* target feature random variable as

$$\psi_{h,U}^T(a_{0:h-1}) \triangleq U\big(\phi_T(s_h)\big), \tag{2}$$

where $(s_0, \ldots, s_h)$ is the state sequence obtained by executing $a_0, \ldots, a_{h-1}$ in the respective MDP.

Algorithm 1 Learning to Undo with LUMOS

Input: source $\mathcal{M}_S$, target $\mathcal{M}_T$, trajectories n, iterations N, moments M, truncated horizon $H^\dagger$
Initialize undo map parameters ϕ.
for $i \in [N]$ **do**
  Sample an action sequence $a_{0:H^\dagger-1}$ uniformly at random.
  ▷ Exactly compute source feature mean and marginal central moments.
  Compute mean $\mu_S(h) \leftarrow \mathbb{E}[\psi_h^S(a_{0:h-1})], \forall h \in [H^\dagger]$.
  Compute moments $c_{k,S}(h) \leftarrow \mathbb{E}\big[(\psi_h^S(a_{0:h-1}) - \mu_S(h))^{\odot k}\big], \forall k \in [M], \forall h \in [H^\dagger]$.
  ▷ Estimate *undone* target feature mean and marginal central moments.
  Collect n trajectories in $\mathcal{M}_T$ using action sequence $a_{0:H^\dagger-1}$.
  for $h \in [H^\dagger]$ **do**
    For each trajectory j, compute *undone* target features $\psi_{h,U_\phi}^{T,(j)}(a_{0:h-1})$.
    Compute mean $\hat{\mu}_T(h) \leftarrow \frac{1}{n}\sum_{j=1}^n \psi_{h,U_\phi}^{T,(j)}(a_{0:h-1})$.
    Compute moments $\hat{c}_{k,T}(h) \leftarrow \frac{1}{n}\sum_{j=1}^n \big(\psi_{h,U_\phi}^{T,(j)}(a_{0:h-1}) - \hat{\mu}_T(h)\big)^{\odot k}, \forall k \in [M]$.
  ▷ Minimize empirical undo discrepancy.
  Update ϕ to minimize $\sum_{h=0}^{H^\dagger-1}\Big[\|\mu_S(h) - \hat{\mu}_T(h)\|_2 + \sum_{k=2}^M \|c_{k,S}(h) - \hat{c}_{k,T}(h)\|_2\Big]$.
Output: undo map parameters ϕ

By the definition of the true undo map $U_\star$, we have for all $h \in [H]$,

$$\psi_h^S(a_{0:h-1}) \overset{d}{=} \psi_{h,U_\star}^T(a_{0:h-1}), \quad (3)$$

where $\overset{d}{=}$ denotes equality in distribution, which means that applying $U_\star$ to the target observations recovers exactly the source distribution, conditioned on any open-loop action sequence $a_{0:h-1}$.

Motivated by Eq. 3, our goal is to evaluate a candidate undo map U by measuring how closely the *undone* target features match the source features in distribution, under open-loop rollouts. To quantify distributional mismatch, we use the Central Moment Discrepancy (CMD) metric [Zellinger et al., 2017], which compares distributions via their means and central moments.

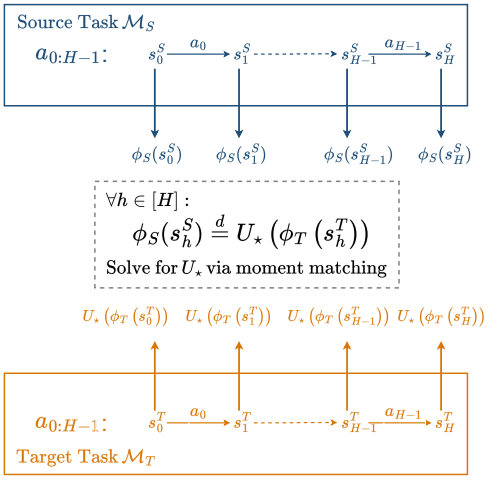


Figure 3: Overview of LUMOS. We match state feature statistics between source and *undone* target trajectories.

Definition 3.2 (Central Moment Discrepancy (CMD) [Zellinger et al., 2017])**.** Let p and q be probability distributions supported on a compact domain $[a,b]^N$, and let $X \sim p$ and $Y \sim q$ be bounded random vectors. The Central Moment Discrepancy (CMD) between p and q is defined as

$$\mathrm{CMD}(p,q) \triangleq \frac{1}{|b-a|}\big\|\mathbb{E}[X] - \mathbb{E}[Y]\big\|_2 + \sum_{k=2}^{\infty}\frac{1}{|b-a|^k}\big\|c_k(X) - c_k(Y)\big\|_2, \quad (4)$$

where $c_k(\cdot)$ denotes the k-th order central moment vector.

$\mathrm{CMD}(p,q) = 0$ iff $p = q$. Moreover, if $\mathrm{CMD}(p_n,p) \to 0$, then $p_n \xrightarrow{d} p$, where $\xrightarrow{d}$ denotes convergence in distribution [Zellinger et al., 2017].

For notational convenience, given random variables X and Y, we write $d_{\mathrm{CM}}(X,Y)$ to denote the CMD between the distributions induced by X and Y. We now use this to define a measure for the quality of an arbitrary undo map.

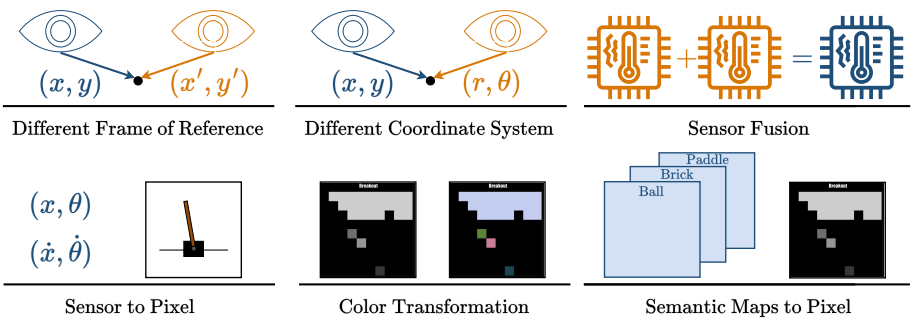


Figure 4: We evaluate LUMOS under structured transformations ranging from geometric and sensor-level changes to gaps in the level of abstraction (e.g., semantic maps in the source and pixels in the target), where the source task is often easier to learn than the target.

Definition 3.3 (Undo Discrepancy)**.** For an undo map U, we define its discrepancy as

$$\mathcal{D}(U) \triangleq \mathbb{E}_{a_{0:H-1}}\left[\sum_{h=0}^{H-1} d_{\mathrm{CM}}\Big(\psi_h^S(a_{0:h-1}), \psi_{h,U}^T(a_{0:h-1})\Big)\right], \tag{5}$$

where $\psi_h^S(a_{0:h-1})$ and $\psi_{h,U}^T(a_{0:h-1})$ denote the source and *undone* target feature random variables at timestep h, and the expectation is over uniformly sampled action sequences.

Thus, minimizing $\mathcal{D}(U)$ is equivalent to matching the *undone* feature distributions in the target to the corresponding feature distributions in the source at each timestep for open-loop sequences of actions.

3.2 Learning the Undo Map

With a distributional measure of undo map quality defined, we now describe how to learn the map.

Empirical Undo Discrepancy. To empirically estimate $\mathcal{D}(U)$ from samples, we approximate CMD using only the first M moments and compute empirical marginal central moments across state feature coordinates. Concretely, given samples $\{x_i\}_{i=1}^n \sim p$, the k-th order empirical marginal central moment vector, $\hat{c}_k(p)$, is given by $\frac{1}{n}\sum_{i=1}^n (x_i - \hat{\mu}_p)^{\odot k}$, where $(\cdot)^{\odot k}$ denotes element-wise exponentiation and $\hat{\mu}_p = \frac{1}{n}\sum_{i=1}^n x_i$ is the empirical mean.

Learning Algorithm. We parameterize the undo map as U_ϕ and optimize its parameters ϕ to match state feature distributions in the source and the *undone* target. More concretely, we minimize the empirical undo discrepancy $\widehat{\mathcal{D}}(U_\phi)$, defined via a truncated empirical CMD estimator using the first M marginal central moments. Our algorithm samples multiple action sequences and optimizes these moments over the resulting rollouts. As samples from the source MDP $\mathcal{M}_S$ are free, we can compute these moments in the source exactly, while they are empirically estimated using samples from the target MDP $\mathcal{M}_T$. Moreover, we use a truncated horizon $H^\dagger < H$ to allow more action sequences to be sampled for training under a given budget of samples; we justify this choice in Section 4.

Algorithm 1 gives the pseudocode for our algorithm, LUMOS. In summary, we iteratively sample an action sequence, compute marginal central moments of source state features, estimate marginal central moments of the corresponding *undone* target state features via n trajectories, and update ϕ to minimize the empirical discrepancy. In practice, the source expectations are also estimated from samples. We then transfer the source policy to the target by composing it with the learned undo map.

3.3 Theoretical Analysis

In this section, we show that LUMOS is more sample efficient than learning from scratch. For this analysis, we specialize our formalism to state-action features used to represent the optimal Q-function. We assume that the source and target state-action features are related by an undo map. Recovering it allows the source optimal Q-function (and therefore its greedy policy) to be transferred to the target.

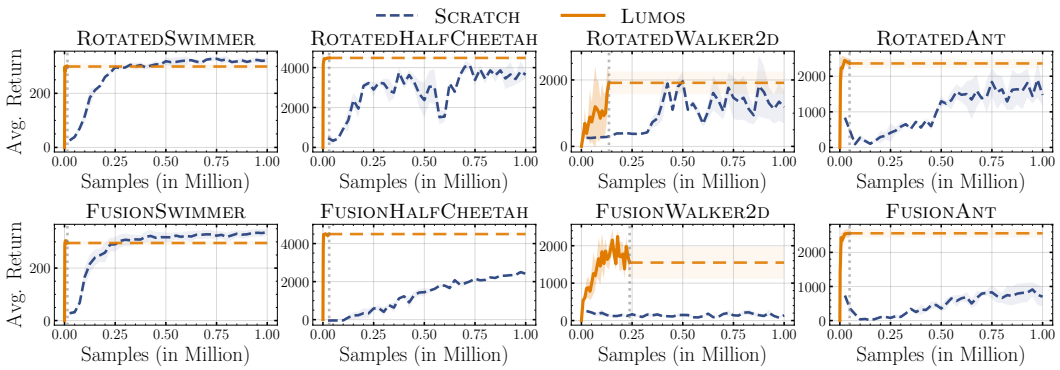


Figure 5: Comparison of avg. return (standard deviation over five random seeds) for varying budgets of target samples. Transfer learning via LUMOS consistently achieves significantly better sample efficiency than learning from scratch.

We first list the assumptions used in our analysis. The source and target MDPs have feature maps $\phi_S : \mathcal{S}^S \times \mathcal{A} \to \mathbb{R}^{d_S}$ and $\phi_T : \mathcal{S}^T \times \mathcal{A} \to \mathbb{R}^{d_T}$ that represent state–action pairs as d_S and d_T-dimensional vectors. Let $\Phi_S \in \mathbb{R}^{|\mathcal{S}^S \times \mathcal{A}| \times d_S}$ and $\Phi_T \in \mathbb{R}^{|\mathcal{S}^T \times \mathcal{A}| \times d_T}$ denote the corresponding feature matrices, with rows $\phi_S(s,a)$ and $\phi_T(s,a)$. We assume that the feature vectors in both MDPs satisfy $\|\phi(s,a)\|_2 \leq 1$, and that rewards are bounded within the range $[-1,1]$. We further assume that the source is linearly-$Q^\star$ realizable.

Assumption 3.4 (Linear-$Q^\star$ realizability in $\mathcal{M}_S$ [Weisz et al., 2021])**.** There exists a vector $\theta_S^\star \in \mathbb{R}^{d_S}$ such that for every state–action pair $(s,a) \in \mathcal{S}^S \times \mathcal{A}$ in the source MDP $\mathcal{M}_S$,

$$Q_S^\star(s,a) \;=\; \langle \phi_S(s,a), \theta_S^\star \rangle.$$

In this setup, given an action sequence $a_{0:h-1}$, we define $\psi_h(a_{0:h-1})$ to be the state-action feature random variable at timestep $h-1$:

$$\psi_h(a_{0:h-1}) \triangleq \phi(s_{h-1}, a_{h-1}). \tag{6}$$

The source and the target are exactly the same except for the state-action features, which are related via a linear undo map $U_\star$: $\Phi_S = \Phi_T U_\star^T$. We assume that the ℓ_2 norm of each row of $U_\star$ is upper bounded by a constant. In the source, the learner has access to the parameter vector $\theta_S^\star$ for the state-action values of the optimal policy, and an oracle $\mathcal{O}$ for the expected state-action features.

Assumption 3.5 (Oracle for expected state-action features in $\mathcal{M}_S$)**.** We denote by $\mathcal{O}$ an oracle that when queried with $a_{0:h-1}$ and $h \in [H]$, outputs $\bar{\psi}_h^S(a_{0:h-1}) \triangleq \mathbb{E}\left[\psi_h^S(a_{0:h-1})\right]$.

We would like to note that this assumption is only for convenience; the same analysis holds when the source expectations are estimated empirically via random-design linear regression arguments.

Since the source and target MDPs differ only through a linear transformation of the state-action features, the expected state-action features in the source and the target (denoted by $\bar{\psi}_h^S(a_{0:h-1})$ and $\bar{\psi}_h^T(a_{0:h-1})$, respectively) are related via the undo map $U_\star$ as follows:

$$\bar{\psi}_h^S(a_{0:h-1}) = U_\star \bar{\psi}_h^T(a_{0:h-1}),\ \forall h \in [H]. \tag{7}$$

Remark. When the undo map is linear, higher-order central moments of the transformed features are completely determined by the first-order statistics through the same linear operator. Therefore, matching higher-order moments provides no additional constraints beyond mean matching.

Learning Algorithm. Our learning algorithm first queries $\mathcal{O}$ to obtain $\bar{\psi}_h^S(a_{0:h-1})$ in $\mathcal{M}_S$, for all $h \in [H]$. Next, it rolls out n trajectories in the target using the open-loop action sequence $a_{0:H-1}$. It then views the state-action features $\psi_h^{T,(i)}(a_{0:h-1})$ (from the i-th sample trajectory) as noisy features for $\bar{\psi}_h^S(a_{0:h-1})$. Subsequently, it estimates $U_\star$ with $\widehat{U}$ via least squares.

Policy Transfer. From Eq. 7, the relation $Q_T^\star = \Phi_T U_\star^T \theta_S^\star$ holds. Thus, a natural way to evaluate the output $\widehat{U}$ of LUMOS is to bound the performance difference with respect to the optimal policy in the target, if we act greedily with respect to $\Phi_T \widehat{U}^T \theta_S^\star$. Let $\hat{\pi}$ denote this policy.

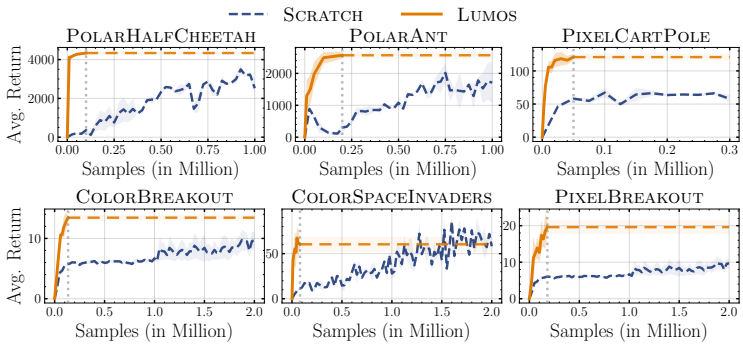


Figure 6: Comparison of avg. return (standard deviation over five random seeds) for varying budgets of target samples. Transfer learning via LUMOS consistently achieves significantly better sample efficiency than learning from scratch.

We show that our algorithm has the following guarantee; the proof is provided in Appendix B.

Theorem 3.6 (Performance Difference Bound for LUMOS)**.** *Define the covariance matrix*

$$\Sigma \triangleq \frac{1}{H}\sum_{h=1}^{H} \bar{\psi}_h^T(a_{0:h-1})\bar{\psi}_h^T(a_{0:h-1})^T,$$

and let $\lambda_{\min}$ *denote* $\lambda_{\min}(\Sigma)$.

For all $n \geq \frac{6}{H\lambda_{\min}}(\log\frac{3d_T d_S}{\delta})$*, with probability at least* $1-\delta$*, the performance difference satisfies*

$$\|V_T^\star - V_T^{\hat{\pi}}\|_\infty \leq \widetilde{O}\left(\sqrt{\frac{H}{n\lambda_{\min}}}\sqrt{d_T + 2\log\left(\frac{3d_S}{\delta}\right)}\right).$$

The theorem shows that if the *open-loop* action sequence $a_{0:H-1}$ is *good*, i.e., $\lambda_{\min}(\Sigma)$ is sufficiently large (on the order of $1/d_T$), then LUMOS achieves a sample complexity upper bound that scales as $\widetilde{O}(d_T\sqrt{H})$. This improves upon the lower bound for linearly-$Q^\star$ realizable MDPs, which scales exponentially with the feature dimension d_T or the horizon H [Weisz et al., 2021].

Remark on Linear MDPs. A special case of MDPs with linear-$Q^\star$ realizability is linear MDPs [Jin et al., 2020]. In this setting, the minimax lower bound has a $d_T H^{3/2}$ dependence on d_T and H [He et al., 2023], whereas LUMOS enjoys a $d_T\sqrt{H}$ dependence, up to logarithmic factors.

4 Experimental Evaluation

In this section, we empirically evaluate our method to address the following research questions: (i) Does transfer via LUMOS outperform learning from scratch? Concretely, does learning the undo map $U_\star$ yield better target policies than single-task training? (ii) How does LUMOS compare with transfer learning baselines that assume reward feedback in the target, which is an orthogonal setting to ours? (iii) Does matching higher-order moments contribute to performance gains? (iv) How should the training budget be allocated between the number of iterations N and the truncated horizon $H^\dagger$?

We begin by describing the transfer settings. Next, we provide an overview of the training process, followed by empirical results. We additionally analyze the robustness of LUMOS in Appendix E.

4.1 Environments

We evaluate our method on a suite of continuous control tasks and Atari games that are challenging to learn from scratch. As source tasks, we consider standard MuJoCo environments: SWIMMER, HALFCHEETAH, WALKER2D, ANT, and CARTPOLE, as well as Atari games: BREAKOUT and SPACEINVADERS. We construct corresponding target tasks by applying structured transformations to the state space. Detailed descriptions are provided in Appendix C.

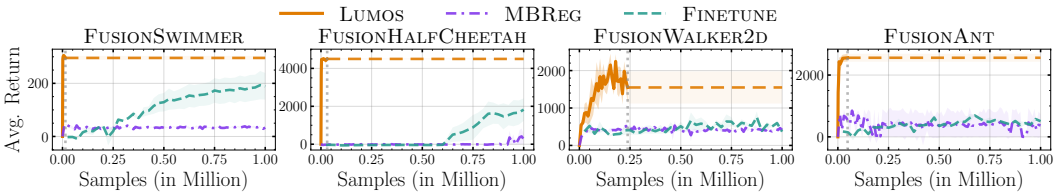


Figure 7: Comparison of avg. return (standard deviation over five random seeds). LUMOS consistently achieves better sample efficiency than transfer baselines that require rewards in the target.

4.2 Training Procedure

We use the `rl-baselines3-zoo` framework [Raffin, 2020] to train both the source policies and the target policies used in the learning-from-scratch baseline, with PPO [Schulman et al., 2017] as the learning algorithm. For each environment, we adopt the tuned hyperparameters provided in the framework. Our Atari experiments are based on the `MinAtar` benchmark [Young and Tian, 2019].

To estimate the state feature means and marginal central moments for undo discrepancy, we use a large number of samples from the source MDP to obtain accurate statistics, while using fewer samples from the target. We employ multiple short action sequences for good coverage of the features.

The undo map is parameterized by a neural network and trained via LUMOS. Additional implementation details are provided in Appendix D.

4.3 Results

In this section, we first compare our method, LUMOS, against learning the target policy from scratch (listed as SCRATCH). Next, we evaluate the contribution of higher-order moments to performance gains. Finally, we examine the effect of using a truncated horizon $H^\dagger$.

Results with LUMOS. We evaluate LUMOS in transfer settings that go beyond the regime of our theoretical analysis. Figures 5 and 6 report the average return (with standard deviation over five random seeds) as a function of the target sampling budget. We stop LUMOS early upon reaching near-source performance, after which the dashed line extends the final performance to larger budgets.

Across all tasks, LUMOS achieves consistent and substantial improvements over single-task training (SCRATCH). The FUSION and PIXEL targets are especially challenging for learning from scratch, due to the increased observation dimensionality and the fact that useful information must be recovered through structured aggregation (across sensors in FUSION, or from high-dimensional visual inputs in PIXEL). In contrast, LUMOS explicitly exploits this shared structure between source and target to achieve near-optimal performance with few target samples.

Overall, these results provide empirical evidence that undo-map-based transfer leads to sample-efficiency gains in RL under structured state-space transformations.

Comparison with transfer RL baselines. While our setup assumes no reward feedback in the target, for completeness we compare LUMOS against transfer RL baselines that operate with reward feedback in the target. Figure 7 includes a standard transfer baseline (FINETUNE) that initializes the target policy from the source policy (except for the input layer when dimensions differ). We also evaluate MBREG [Sun et al., 2022], a baseline that learns a reward and dynamics model over latent states in the source and uses it for model regularization in the target. Despite the favorable conditions afforded to these baselines (namely, access to reward feedback in the target), LUMOS consistently outperforms them across a variety of environments.

Ablation w.r.t. the number of moments. We use PIXELCARTPOLE to study the effect of matching higher-order central moments when learning the undo map. Specifically, we run LUMOS with $M = 1$ (mean matching only) and $M = 2$ (including second-order moments). Figure 8(a) shows that using higher-order moments yields clear performance improvements.

Ablation w.r.t. the truncated horizon. We use PIXELCARTPOLE to justify our algorithmic choice of truncating trajectories at horizon $H^\dagger$. Keeping the overall sampling budget fixed, we vary the allocation between the number of sampled action sequences N and their length $H^\dagger$. Concretely, we consider the configurations $(N, H^\dagger) \in \{(5, 500), (50, 50), (500, 5)\}$, with the product $NH^\dagger$ kept constant. Figure 8(b) indicates that moderately short sequences perform best: using shorter sequences

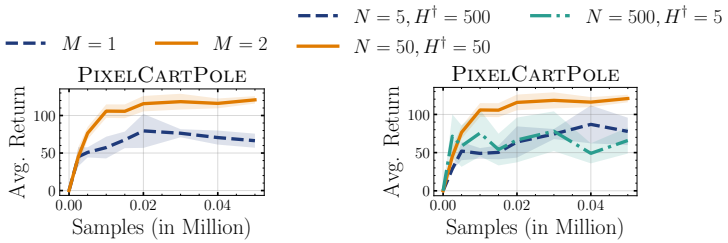


Figure 8: Ablation results. **(a)** Effect of the number of matched moments M. **(b)** Effect of allocating a fixed sampling budget between the number of action sequences N and their truncated horizon $H^\dagger$.

improves performance relative to very long rollouts (e.g., $(50, 50)$ vs. $(5, 500)$), while overly short horizons can hurt due to potentially insufficient state-space coverage (e.g., $(500, 5)$).

5 Related Work

Transfer Learning. Empirical methods for transfer learning in RL often lack provable guarantees or rely on application-specific assumptions that do not generalize. For instance, Sun et al. [2022] assume shared latent dynamics across observation spaces and that target observations can be transformed into the source domain; they do not learn this mapping explicitly and offer no theoretical guarantees on transfer performance. Similarly, Watahiki et al. [2024] assume a shared latent MDP structure between tasks but their method is not theoretically grounded. Other works, such as Agarwal et al. [2022], study transfer through reusing prior policies across design iterations, which is orthogonal to our setting. Methods like Chen et al. [2024], Yi et al. [2023] rely on very specific visual or object-level assumptions, which limits their general applicability. The notion of an *undo* map to reverse state space transformations in reinforcement learning was first introduced in Gupta et al. [2022], where the problem is approached through a distributional lens by aligning trajectories across the source and target tasks. However, the method does not scale beyond toy tabular environments. Orthogonal to our setup, works such as Qu et al. [2025], Wagenmaker et al. [2024] study transfer under dynamics shift.

Representational Transfer in Low-Rank MDPs. Several works have studied representational transfer under low-rank or linear MDP assumptions. For instance, Agarwal et al. [2023], Cheng et al. [2022] use reward-free exploration in the source task to learn a good representation for the target. Sam et al. [2024] extend this to a more general low-rank setting. Lu et al. [2021] learn a linear representation using least squares for multitask linear MDPs, and Ishfaq et al. [2024] consider the offline multitask case. Chai et al. [2025] provide guarantees for transfer Q-learning in the composite MDP setting, but their method does not scale empirically and only works for finite state-action spaces.

In contrast, our work approaches transfer RL through a more principled lens by making concrete assumptions about the structural relationship between the source and target tasks. We propose an algorithm that explicitly learns a mapping between their observation spaces by framing the problem as supervised learning, and therefore eliminates the need for RL in the target task. Under certain assumptions, this leads to provable improvements in sample complexity over learning from scratch. Moreover, our method is practical and demonstrates strong empirical performance. Essentially, our work takes a step toward more principled and broadly applicable approaches to transfer in RL.

Our goal is not to develop a state-of-the-art transfer RL method competing across benchmarks, but rather to design a theoretically grounded approach that is provably better than learning from scratch when suitable shared structure exists, and is also empirically better than single-task RL.

6 Concluding Discussion

In this paper, we introduced a principled approach to transfer learning in reinforcement learning, where the source and target tasks are related by structured transformations of the state space. By explicitly learning an undo map, our method achieves significant gains in sample efficiency compared to learning target policies from scratch, as demonstrated across a range of challenging continuous control tasks and Atari games. Theoretically, we showed that when the undo map is linear and the source is linearly-$Q^\star$ realizable, our approach achieves strictly better sample complexity than learning from scratch. A promising direction for future work is to study more general notions of task equivalence, such as MDP homomorphisms where the action space may also be transformed. Extending our guarantees beyond the linear realizability assumption is another interesting direction.

Broader impact. Our work is algorithmic, and we do not see any direct societal implications.

Funding. No external funding was received.

Competing Interests. The authors declare no competing interests.

References

Alekh Agarwal, Yuda Song, Wen Sun, Kaiwen Wang, Mengdi Wang, and Xuezhou Zhang. Provable Benefits of Representational Transfer in Reinforcement Learning. *COLT*, 2023.

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc G. Bellemare. Reincarnating Reinforcement Learning: Reusing Prior Computation to Accelerate Progress. *NeurIPS*, 2022.

Steven Bohez, Tim Verbelen, Elias De Coninck, Bert Vankeirsbilck, Pieter Simoens, and Bart Dhoedt. Sensor Fusion for Robot Control through Deep Reinforcement Learning. *IROS*, 2017.

Jinhang Chai, Elynn Chen, and Lin Yang. Transfer Q-Learning with Composite MDP Structures. *ICML*, 2025.

Lawrence Yunliang Chen, Kush Hari, Karthik Dharmarajan, Chenfeng Xu, Quan Vuong, and Ken Goldberg. Mirage: Cross-Embodiment Zero-Shot Policy Transfer with Cross-Painting. *RSS*, 2024.

Yuan Cheng, Songtao Feng, Jing Yang, Hong Zhang, and Yingbin Liang. Provable Benefit of Multitask Representation Learning in Reinforcement Learning. *NeurIPS*, 2022.

Gabriel Dulac-Arnold, Daniel J. Mankowitz, and Todd Hester. Challenges of Real-World Reinforcement Learning. *CoRR*, abs/1904.12901, 2019.

Abhi Gupta, Ted Moskovitz, David Alvarez-Melis, and Aldo Pacchiano. Transfer RL via the Undo Maps Formalism. *CoRR*, abs/2211.14469, 2022.

Jiafan He, Heyang Zhao, Dongruo Zhou, and Quanquan Gu. Nearly Minimax Optimal Reinforcement Learning for Linear Markov Decision Processes. *ICML*, 2023.

Daniel J. Hsu, Sham M. Kakade, and Tong Zhang. Random Design Analysis of Ridge Regression. *Found. Comput. Math.*, 14(3):569–600, 2014.

Edward S. Hu, James Springer, Oleh Rybkin, and Dinesh Jayaraman. Privileged Sensing Scaffolds Reinforcement Learning. *ICLR*, 2024.

Haque Ishfaq, Thanh Nguyen-Tang, Songtao Feng, Raman Arora, Mengdi Wang, Ming Yin, and Doina Precup. Offline Multitask Representation Learning for Reinforcement Learning. *NeurIPS*, 2024.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably Efficient Reinforcement Learning with Linear Function Approximation. *COLT*, 2020.

Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Sci. Robotics*, 4(26), 2019.

Rui Lu, Gao Huang, and Simon S Du. On the Power of Multitask Representation Learning in Linear MDP. *CoRR*, abs/2106.08053, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Chengrui Qu, Laixi Shi, Kishan Panaganti, Pengcheng You, and Adam Wierman. Hybrid Transfer Reinforcement Learning: Provable Sample Efficiency from Shifted-Dynamics Data. *AISTATS*, 2025.

Antonin Raffin. RL Baselines3 Zoo. `https://github.com/DLR-RM/rl-baselines3-zoo`, 2020.

Tyler Sam, Yudong Chen, and Christina Lee Yu. The Limits of Transfer Reinforcement Learning with Latent Low-rank Structure. *NeurIPS*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347, 2017.

Yanchao Sun, Ruijie Zheng, Xiyao Wang, Andrew Cohen, and Furong Huang. Transfer RL across Observation Feature Spaces via Model-Based Regularization. *ICLR*, 2022.

Matthew E. Taylor and Peter Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *JMLR*, 2009.

Andrew Wagenmaker, Kevin Huang, Liyiming Ke, Byron Boots, Kevin Jamieson, and Abhishek Gupta. Overcoming the Sim-to-Real Gap: Leveraging Simulation to Learn to Explore for Real-World RL. *NeurIPS*, 2024.

Hayato Watahiki, Ryo Iwase, Ryosuke Unno, and Yoshimasa Tsuruoka. Cross-Domain Policy Transfer by Representation Alignment via Multi-Domain Behavioral Cloning. *CoLLAs*, 2024.

Gellért Weisz, Philip Amortila, and Csaba Szepesvári. Exponential Lower Bounds for Planning in MDPs With Linearly-Realizable Optimal Action-Value Functions. *ALT*, 2021.

Qi Yi, Rui Zhang, Shaohui Peng, Jiaming Guo, Yunkai Gao, Kaizhao Yuan, Ruizhi Chen, Siming Lan, Xing Hu, Zidong Du, Xishan Zhang, Qi Guo, and Yunji Chen. Online Prototype Alignment for Few-shot Policy Transfer. *ICML*, 2023.

Kenny Young and Tian Tian. MinAtar: An Atari-Inspired Testbed for Thorough and Reproducible Reinforcement Learning Experiments. *CoRR*, abs/1903.03176, 2019.

Werner Zellinger, Thomas Grubinger, Edwin David Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central Moment Discrepancy (CMD) for Domain-Invariant Representation Learning. *ICLR*, 2017.

Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: a Survey. *IEEE SSCI*, 2020.

Henry Zhu, Abhishek Gupta, Aravind Rajeswaran, Sergey Levine, and Vikash Kumar. Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost. *ICRA*, 2019.

Zhuangdi Zhu, Kaixiang Lin, Anil K. Jain, and Jiayu Zhou. Transfer Learning in Deep Reinforcement Learning: A Survey. *IEEE TPAMI*, 2020.

A Table of Contents

In this section, we provide an outline of the contents provided in the paper's appendices.

- Appendix B contains the proof of Theorem 3.6.
- Appendix C describes the environments used in the experimental evaluation.
- Appendix D provides additional implementation details for the experimental evaluation.
- Appendix E provides additional results examining the robustness of LUMOS.

B Proofs

In this section, we provide the proof of Theorem 3.6.

Proof Structure. The overall structure of our proof is:

1. View the estimation of each row of $\widehat{U}$ as a random-design linear regression problem and bound its covariance-weighted estimation error.
2. Use $\lambda_{\min}(\Sigma)$ to convert this covariance-weighted error into a max-norm bound on $\widehat{U} - U_\star$.
3. Use the relation $Q_T^\star = \Phi_T U_\star^\top \theta_S^\star$ to translate undo-map estimation error into error in the approximate target Q-function $\widehat{Q_T} = \Phi_T \widehat{U}^\top \theta_S^\star$.
4. Apply a standard performance bound for the policy greedy with respect to $\widehat{Q_T}$ to obtain the final value-difference guarantee.

We first relate the performance difference between the optimal policy in the target $\mathcal{M}_T$ and the policy that acts greedily with respect to $\Phi_T \widehat{U}^T \theta_S^\star$ to the estimation error in $\widehat{U}$, where $\widehat{U}$ estimates $U_\star$.

Lemma B.1 (Performance Difference)**.** $\|V_T^\star - V_T^{\hat{\pi}}\|_\infty \le 2H \|\widehat{U} - U_\star\|_{\max}$.

Here, $\|\cdot\|_{\max}$ denotes the max norm. To bound the estimation error $\|\widehat{U} - U_\star\|_{\max}$, we use the following per-row guarantee:

Lemma B.2 (Estimation Error Bound)**.** *Let $M[i]$ denote the i-th row of a matrix M. Fix $i \in [d_S]$. Then, for all $n \ge \frac{6}{H\lambda_{\min}}(\log \frac{3d_T}{\delta})$, with probability at least $1-\delta$, the output $\widehat{U}$ of* LUMOS *satisfies:*

$$\left\|\widehat{U}[i] - U_\star[i]\right\|_\Sigma \le \sqrt{\frac{1}{nH}} \sqrt{d_T + 2\log\left(\frac{3}{\delta}\right) + 2\sqrt{d_T \log\left(\frac{3}{\delta}\right)}} + o\left(\sqrt{\frac{1}{nH}}\right).$$

Since Σ is positive semidefinite (PSD), we can relate the Σ-norm to the ℓ_∞ norm via:

$$\|x\|_\infty \le \frac{\|x\|_\Sigma}{\sqrt{\lambda_{\min}}}. \tag{8}$$

Applying a union bound over all $i \in [d_S]$, with per-row failure probability set to δ/d_S, gives a high-probability bound on $\|\widehat{U} - U_\star\|_{\max}$. Substituting this into Lemma B.1 concludes the proof.

Proof of Lemma B.1. Let f denote the approximate target Q-function induced by the estimated undo map: $f(s,a) \triangleq \left\langle \widehat{U}\phi_T(s,a), \theta_S^\star \right\rangle$. For any $s \in \mathcal{S}^T$,

$$\begin{aligned}
V_T^\star(s) - V_T^{\hat{\pi}}(s) &= Q_T^\star(s,\pi^\star(s)) - Q_T^\star(s,\hat{\pi}(s)) + Q_T^\star(s,\hat{\pi}(s)) - Q_T^{\hat{\pi}}(s,\hat{\pi}(s)) \\
&\le Q_T^\star(s,\pi^\star(s)) - f(s,\pi^\star(s)) + f(s,\hat{\pi}(s)) - Q_T^\star(s,\hat{\pi}(s)) \\
&\quad + \mathbb{E}_{s'\sim\mathbb{P}^T(s,\hat{\pi}(s))}\left[V_T^\star(s') - V_T^{\hat{\pi}}(s')\right] \\
&\le 2\|f - Q_T^\star\|_\infty + \mathbb{E}_{s'\sim\mathbb{P}^T(s,\hat{\pi}(s))}\left[V_T^\star(s') - V_T^{\hat{\pi}}(s')\right].
\end{aligned}$$

Unrolling this recursion for H steps,

$$\|V_T^\star - V_T^{\hat{\pi}}\|_\infty \leq 2H\|f - Q_T^\star\|_\infty.$$

Substituting $f = \Phi_T \widehat{U}^\top \theta_S^\star$ and $Q_T^\star = \Phi_T U_\star^\top \theta_S^\star$ completes the proof.

Proof of Lemma B.2. Fix a source-feature coordinate $i \in [d_S]$. Estimating $U_\star[i]$ can be viewed as a random-design linear regression problem in which the target state-action features (of dimensionality d_T) are the covariates and the corresponding expected source-feature coordinate is the response. The regression parameter is $U_\star[i]$, and the covariance matrix is Σ. Hsu et al. [2014] bound the covariance-weighted estimation error of least squares under three conditions.

In our setting:

1. The regression noise is bounded in $[-1, 1]$ and is therefore 1-subgaussian, so we set $\sigma = 1$.
2. The linear model is correctly specified through $\overline{\psi}_h^S = U_\star \overline{\psi}_h^T$. Therefore, there is no approximation bias, and we set $b_0 = 0$.
3. Since the ℓ_2-norm of $\phi_T(s, a)$ is at most 1 and the minimum eigenvalue of Σ is $\lambda_{\min}$, the statistical leverage parameter ρ_0 satisfies $\rho_0 \leq \frac{1}{\sqrt{\lambda_{\min} d_T}}$ by Remark 2 of [Hsu et al., 2014].

Substituting these quantities into Theorem 1 and Remark 9 of [Hsu et al., 2014] completes the proof.

C Environments

C.1 Types of Transformations

Changed frame of reference. Here, we rotate the global coordinate frame in the source tasks, which results in rotated variants of the MuJoCo tasks (e.g., ROTATEDANT). This models geometric discrepancies that arise, for example, from changes in coordinate conventions across simulators.

Sensor fusion. In this setting, multiple sensor modalities are available, which results in sensor-fused variants (e.g., FUSIONANT). Instead of observing a d_S-dimensional vector, the agent receives a $10d_S$-dimensional observation ($10\times$ sensors), which can be combined to recover the original values. This captures settings where sensors differ at deployment.

Changed coordinate system. We construct polar-coordinate variants POLARHALFCHEETAH and POLARANT, where sensor readings are expressed in polar rather than Cartesian coordinates. This captures settings with differing state parameterizations.

Sensor-to-pixel observations. We consider CARTPOLE with sensor observations as the source task and pixel-based observations as the target, which results in PIXELCARTPOLE. This introduces an abstraction gap where the source provides structured low-dimensional inputs, while the target requires learning directly from raw visual observations.

Color transformation. For Atari games, we treat grayscale observations as the source and RGB observations as the target, which results in the variants COLORBREAKOUT and COLORSPACEINVADERS. This setting models rendering shifts at deployment time.

Semantic bitmaps to pixels. Finally, we consider a setting where the source provides object-centric semantic bitmaps, but the target provides raw pixel observations (PIXELBREAKOUT). This represents an abstraction gap where learning from scratch is significantly harder.

C.2 Environment-Specific Details

ROTATEDSWIMMER. This task requires the agent to swim forward through a viscous fluid. We apply a 45° rotation to the (x, y) plane to modify the frame in which the agent's motion is observed. This affects both the heading of the swimmer and its linear velocity vector. The observation is updated accordingly by adjusting the front tip's angle and rotating the (x, y) velocity components.

ROTATEDHALFCHEETAH. This task requires the agent to run forward in a planar environment. The original environment operates in a 2D plane. We perform a 45° rotation on the (x, z) components of the agent's torso position and velocity. The joint states and angular velocities remain unchanged.

ROTATEDWALKER2D. This task requires the agent to walk forward while maintaining balance. Similar to HalfCheetah, we rotate the observation of the agent's torso height and linear velocity in the (x, z) plane by $45°$. This alters how progress and vertical motion are perceived, while the actuator and sensor states are kept intact.

ROTATEDANT. In this task, the agent is a quadruped and must navigate a 2D surface using four legs. We rotate the global (x, y) velocity of the torso by $45°$. When present, external contact forces are also rotated in the same plane. The result is a consistent shift in perceived motion direction across the ant's high-dimensional observation space.

Sensor Fusion. In the fusion environments, instead of observing a d_S-dimensional observation vector, the agent receives a $10 \times d_S$-dimensional observation. More concretely, for each source observation element i, the target task provides 10 measurements of the form: $x_i^{(j)} = w_j \cdot x_i + c_j$, for $j \in [10]$, where the weights w_j sum to 1 and the offsets c_j sum to 0. This simulates a sensor fusion scenario in which the undo map $U_\star$ sums the 10 measurements corresponding to each original dimension to recover the source observation.

POLARHALFCHEETAH. Here, we change the coordinate system of selected observation dimensions from Cartesian to polar coordinates. In particular, pairs of state variables such as the angular velocities of the back thigh and shin are transformed so that their joint (x, y) representation becomes (r, θ), where $r = \sqrt{x^2 + y^2}$ and $\theta = \arctan 2(y, x)$. This alters how the agent perceives limb velocities.

POLARANT. In this task, the angular velocity components of the torso along the x and y axes are represented in polar form. This affects how rotational motion is encoded.

PIXELCARTPOLE. In this task, the agent receives pixel observations instead of low-dimensional sensor states. We extract a cropped 40×60 image centered around the cart, and the observation is the difference between two consecutive frames to capture dynamics. Since the cart's absolute position and velocity cannot be inferred from pixels alone, we mask those variables from the source task to ensure comparability. This setting, inspired by Sun et al. [2022], evaluates whether our method can transfer across a modality shift from sensor states to pixels.

COLORBREAKOUT and COLORSPACEINVADERS. In these tasks, the source provides grayscale observations while the target provides RGB.

PIXELBREAKOUT. In this task, the source provides semantic bitmaps corresponding to the ball, bricks, paddle, and trail, whereas the target provides grayscale pixel observations.

D Implementation Details

In our implementation, if an episode terminates early, we reset the environment but continue the action sequence indexing. This ensures that we can always collect $H^\dagger$ steps from an action sequence. The action sequences used to compute the state feature statistics are either randomly sampled or derived from the source policy π_S. Concretely, we perform a rollout of π_S in the source MDP and record the action sequence taken by the policy. We then use this recorded sequence. One can potentially include both kinds of action sequences, which makes this procedure task-independent and effectively combines the benefits of both sampling strategies.

Rotation and Fusion Tasks. The undo map is a single linear layer, and we use LUMOS with $M{=}1$ (learning rate 0.001, weight decay 0.0001, 10,000 epochs). For SWIMMER and WALKER2D, we use randomly sampled action sequences of lengths 8 and 17, respectively, with 200 and 100 source trajectories and 3 target trajectories per sequence. For HALFCHEETAH and ANT, the action sequences are obtained from the source policy (with lengths 17 and 27, respectively, and 100 and 60 source trajectories and 3 target trajectories per sequence). We use up to 600 action sequences.

Polar Tasks. The undo map is an MLP with two hidden layers (64 units per layer), and we use LUMOS with $M{=}1$ (learning rate 0.001, weight decay 0.0001, 10,000 epochs). We obtain 100 action sequences of length 100 from the source policy. Target statistics are computed from up to 10 (POLARHALFCHEETAH) and 20 (POLARANT) trajectories per sequence.

Sensor-to-Pixel Task. The undo map is an MLP with one hidden layer (16 units), and we use LUMOS with $M{=}2$ (learning rate 0.001, weight decay 0.0001, 10,000 epochs). We use 50 randomly sampled action sequences of length 50. Target statistics are estimated with up to 20 trajectories per sequence.

Color Tasks. The undo map is a single linear layer, and we use LUMOS with $M{=}1$ (learning rate 0.01, ℓ_1 penalty of 0.001, $3{,}000$ epochs). We use 10 sequences of length 300 (COLORBREAKOUT) or 100 (COLORSPACEINVADERS), obtained from the source policy. Target statistics are estimated with up to 50 (COLORBREAKOUT) and 100 (COLORSPACEINVADERS) trajectories per sequence.

Bitmaps-to-Pixel Task. The undo map is an MLP with one hidden layer (2048 units), and we use LUMOS with $M{=}2$ (learning rate 0.001, weight decay 0.0001, $2{,}000$ epochs). Here, we use 20 source-policy action sequences of length 200, with up to 50 target trajectories per sequence.

More details are provided in the codebase. For the transfer baselines, we use the default hyperparameters provided in the original codebases.

Compute resources. All experiments were conducted on nodes of a shared cluster with Intel Xeon Gold 6426Y CPUs (up to 4.1 GHz, 250 GB RAM per node) and NVIDIA L40S GPUs (46 GB VRAM). Each experiment took approximately 5 to 15 minutes per seed.

E Robustness Analysis

We conduct additional robustness experiments on PIXELCARTPOLE using five random seeds. PIXELCARTPOLE is a nontrivial transfer setting because the source policy operates on low-dimensional sensor observations, whereas the target provides only high-dimensional rendered pixel observations.

We consider two stochastic observation corruptions for which an exact deterministic (memoryless) undo map generally does not exist.

1. **Random Pixel Masking:** Each target pixel is independently set to zero with probability p_m.
2. **Sticky Frames:** The previous observation is emitted instead of the current with probability p_s.

Table 1 reports the results at multiple target-sample budgets for the individual corruptions, as well as for a combined stochastic corruption consisting of pixel masking [$p_m = 0.05$] and sticky frames [$p_s = 0.01$]. All results are averaged over five random seeds.

Table 1: Average return (standard deviation over five random seeds) on PIXELCARTPOLE under stochastic observation corruptions at different target-sample budgets.

	Target Samples			
Setting	12.5k	25k	37.5k	50k
Exact transformation	86.12 ± 18.20	114.20 ± 6.31	115.08 ± 5.85	118.96 ± 3.82
Pixel masking [$p_m = 0.05$]	79.08 ± 11.51	99.96 ± 13.08	99.52 ± 7.00	107.28 ± 13.02
Pixel masking [$p_m = 0.10$]	70.00 ± 20.55	76.64 ± 9.76	87.56 ± 8.78	91.36 ± 13.63
Sticky frames [$p_s = 0.01$]	90.56 ± 9.67	95.76 ± 16.62	104.12 ± 9.05	111.76 ± 5.75
Sticky frames [$p_s = 0.10$]	48.04 ± 13.28	50.80 ± 14.76	49.40 ± 9.05	57.68 ± 10.52
Combined [$p_m = 0.05$, $p_s = 0.01$]	74.04 ± 17.10	91.92 ± 27.11	98.88 ± 13.22	98.80 ± 19.18

Under 5% pixel masking and 1% sticky frames, LUMOS remains close to its performance under the exact transformation. Performance decreases as the corruption becomes more severe with 10% pixel masking or sticky frames. Sticky frames lead to a larger degradation because the entire current observation is occasionally replaced by stale information.

The results show that performance generally improves with additional target samples. Combining the two (mild) stochastic corruptions reduces performance relative to individual corruptions. However, it does not result in a drastic degradation of performance; LUMOS reaches an average return of 98.80 ± 19.18 after 50k samples from the target.

Overall, these results show that LUMOS remains effective under moderate stochastic observation corruption and degrades as the exact undo-map assumption is increasingly violated.

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: Yes, they do.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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 generalize 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 discuss that in the section on concluding discussion.

   Guidelines:

   - The answer [N/A] 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: Yes, we provide the full proof in the appendices.

Guidelines:

- The answer [N/A] 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: Yes, we provide additional information as well in the appendices.

   Guidelines:

   - The answer [N/A] 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: A link to the codebase is provided in the abstract.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, we also provide additional details in the appendices. Some details can be obtained from the codebase, too.

   Guidelines:

   - The answer [N/A] 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: Yes, we do report standard deviation across runs.

   Guidelines:

   - The answer [N/A] 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: Yes, we provide this information in the appendices.

   Guidelines:

   - The answer [N/A] 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: Yes, it does.

   Guidelines:

   - The answer [N/A] 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: [Yes]

    Justification: Yes, it does.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper poses no such risks.

    Guidelines:

    - The answer [N/A] 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: Yes, they are.

    Guidelines:

    - The answer [N/A] 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: Yes, they are.

    Guidelines:

    - The answer [N/A] 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: [N/A]

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

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: Not applicable.

    Guidelines:

    - The answer [N/A] 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.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: Only used as a writing aid.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.