TRANSFER RL ACROSS OBSERVATION FEATURE SPACES VIA MODEL-BASED REGULARIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In many reinforcement learning (RL) applications, the observation space is specified by human developers and restricted by physical realizations, and may thus be subject to dramatic changes over time (e.g. increased number of observable features). However, when the observation space changes, the previous policy will likely fail due to the mismatch of input features, and another policy must be trained from scratch, which is inefficient in terms of computation and sample complexity. Following theoretical insights, we propose a novel algorithm which extracts the latent-space dynamics in the source task, and transfers the dynamics model to the target task to use as a model-based regularizer. Our algorithm works for drastic changes of observation space (e.g. from vector-based observation to image-based observation), without any inter-task mapping or any prior knowledge of the target task. Empirical results show that our algorithm significantly improves the efficiency and stability of learning in the target task.

1 INTRODUCTION

Deep Reinforcement Learning (DRL) has the potential to be used in many large-scale applications which are under rapid evolution and development, such as robotics, gaming and automotives. Concretely, a rapidly evolving environment may be subject to significant changes in the transition dynamics, reward function, and observation space. Therefore, in order for DRL to be a viable solution, algorithms must be able to efficiently cope with evolving environment configurations. Transfer learning promises to address this by providing efficient means of transferring knowledge from a learned task (*source task*) to a related, new task (*target task*).

Most existing transfer RL works (Taylor & Stone, 2009; Zhu et al., 2020) focus on tasks with the same state-action space but different dynamics/reward, such that the source policy can adapt to the target task by learning both task-specific and task-agnostic skills. However, these approaches do not apply to the case where the observation space changes while the dynamics remain similar, which is often the case as humans modify how the agent observes the world (e.g., upgrading a sensor).

Observation change is common in practice due to hardware upgrading, data restriction or curriculum design. For example, the absolute positions of objects in a source task may be replaced by the relative positions between the agent and the objects in a target task. The change may even be drastic, as shown in a motivating example in Figure 1, the vector-based observation (the x-y coordinates of the agent and the goal locations) in a source task (e.g., in a simulator) may be changed to a much higher dimensional image-based observation (a top-down view of the whole maze) in a target-task (e.g., in a real maze environment). In these cases, existing methods may have to discard the old policy and train a new policy from scratch which is inefficient in both time and sample complexity, even though the underlying dynamics are similar. Therefore, there is a crucial need for a technique that transfers knowledge across tasks with similar dynamics but different observation spaces.

In this paper, we aim to fill the gap and propose a new algorithm that can automatically transfer knowledge from the old environment to facilitate learning in a new environment with a different observation space. In order to meet more practical needs, we focus on the challenging setting where the observation change is: **(1) unpredictable** (there is no prior knowledge about how the observations change), **(2) drastic** (the source and target tasks have significantly different observation feature spaces, e.g., vector to image), and **(3) irretrievable** (once the change happens, it is impossible to query the source task, so that the agent can not interact with both environments simultaneously).

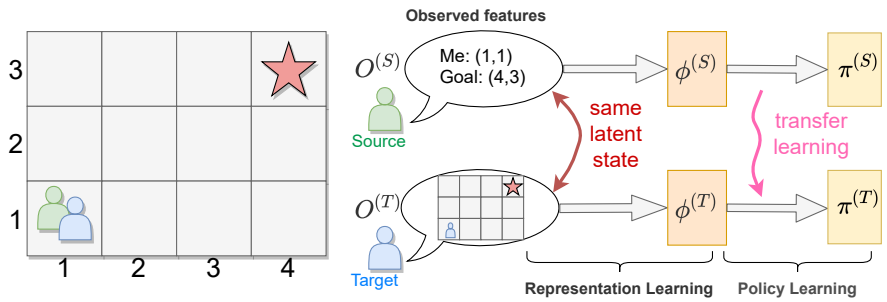


Figure 1: An example of the transfer problem with changed observation space. The source-task agent observes the x-y coordinates of itself and the goal, while the target-task agent observes a top-down view/image of the whole maze. The two observation spaces are drastically different, but the two tasks are structurally similar. Our goal is to transfer knowledge from the source task to accelerate learning in the target task, without knowing or learning any inter-task mapping.

Note that this setting is different from many prior works which require an inter-task mapping as prior knowledge (Taylor et al., 2007; Mann & Choe, 2013), or are designed for the addition or removal of individual observation features (Raiman et al., 2019).

To remedy the above challenges and achieve knowledge transfer, we make a key observation that, if only the observation features change, the source and target tasks share the same latent space and dynamics (e.g. in Figure 1, $\mathcal{O}^{(S)}$ and $\mathcal{O}^{(T)}$ can be associated to the same latent state). Therefore, we first disentangle representation learning from policy learning, and then accelerate the target-task agent by regularizing the representation learning process with the latent dynamics model learned in the source task. We show by theoretical analysis and empirical evaluation that the target task can be learned more efficiently with our proposed transfer learning method than from scratch.

Summary of Contributions. (1) To the best of our knowledge, we are the first to discuss the transfer problem where the source and target tasks have drastically different observation feature spaces, and there is no prior knowledge of an inter-task mapping. (2) We theoretically characterize what constitutes a "good representation" and analyze the sufficient conditions the representation should satisfy. (3) Theoretical analysis shows that a model-based regularizer enables efficient representation learning in the target task. Based on this, we propose a novel algorithm that automatically transfers knowledge across observation representations. (4) Experiments in 5 environments show that our proposed algorithm significantly improves the learning performance of RL agents in the target task.

2 Preliminaries and Background

Basic Notations and Background of RL An RL task can be modeled by a Markov Decision Process (MDP) (Puterman, 2014), defined as a tuple $M = \langle \mathcal{O}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{O}$ is the state/observation space, $\mathcal{A}$ is the action space, P is the transition kernel, R is the reward function and γ is the discount factor. At timestep t, the agent observes state o_t, takes action a_t based on its *policy* $\pi : \mathcal{O} \to \Delta(\mathcal{A})$(where $\Delta(\cdot)$ denotes the space of probability distributions), and receives reward $r_t = R(o_t, a_t)$. The environment then proceeds to the next state $o_{t+1} \sim P(\cdot|o_t, a_t)$.

The goal of an RL algorithm is to find a policy π in the policy space Π with the highest cumulative reward, which is characterized by the state value function V or the state-action value Q. The *value* of a policy π for a state $o \in \mathcal{O}$ is defined as $V^\pi(o) = \mathbb{E}_{\pi,P}[\sum_{t=0}^{\infty} \gamma^t r_t | o_0 = o]$. The *Q value* of a policy π for a state-action pair $(o, a) \in \mathcal{O} \times \mathcal{A}$ is defined as $Q^\pi(o, a) = \mathbb{E}_{\pi,P}[\sum_{t=0}^{\infty} \gamma^t r_t | o_0 = o, a_0 = a]$. For any policy π, its Q value Q^π is the unique fixed point of the Bellman operator

$$(\mathcal{T}^\pi Q)(o, a) = \mathbb{E}_{o' \sim P(o,a), a' \sim \pi(o')}[R(o, a) + \gamma Q(o', a')] \tag{1}$$

The optimal policy can be found by *policy iteration* (Howard, 1960), where one starts from an initial policy π_0 and repeats policy evaluation and policy improvement. More specifically, at iteration k, the algorithm evaluates Q_{π_k} via Equation (1), then improves the policy by $\pi_{k+1}(o) := \operatorname{argmax}_{a \in \mathcal{A}} Q_{\pi_k}(o, a), \forall o \in \mathcal{O}$. It is well-known that the policy iteration algorithm converges to the optimal policy under mild conditions (Puterman, 2014). When the dynamics P and R are unknown, reinforcement learning algorithms use interaction samples from the environment to approximately

solve $\hat{Q}_{\pi_k}$. Prior works (Bertsekas & Tsitsiklis, 1996; Munos, 2005) have shown that if the approximation error is bounded by a small constant, the performance of π_k as $k \to \infty$ is guaranteed to be close to the optimal policy value Q_{π^*}, which we also denote as Q^*.

Representation Learning in RL Real-world applications usually have large observation spaces for which function approximation is needed to learn the value or the policy. However, directly learning a policy over the entire observation space could be difficult, as there is usually redundant information in the observation inputs. A common solution is to map the large-scale observation into a smaller representation space via a *non-linear encoder* (also called a *representation mapping*) $\phi : \mathcal{O} \to \mathbb{R}^d$, where d is the representation dimension, and then learn the policy/value function over the representation space $\phi(\mathcal{O})$. In DRL, the encoder and the policy/value are usually jointly learned.

3 Problem Setup: Transfer Across Different Observation Spaces

We aim to transfer knowledge learned from a source MDP to a target MDP, whose observation spaces are different while dynamics are structurally similar. Denote the source MDP as $\mathcal{M}^{(S)} = \langle \mathcal{O}^{(S)}, \mathcal{A}, P^{(S)}, R^{(S)}, \gamma \rangle$, and the target MDP as $\mathcal{M}^{(T)} = \langle \mathcal{O}^{(T)}, \mathcal{A}, P^{(T)}, R^{(T)}, \gamma \rangle$. Note that $\mathcal{O}^{(S)}$ and $\mathcal{O}^{(T)}$ can be significantly different, such as $\mathcal{O}^{(S)}$ being a low-dimensional vector space and $\mathcal{O}^{(T)}$ being a high-dimensional pixel space, which is challenging for policy transfer since the source target policy have different input shapes and would typically be very different architecturally.

In this work, as motivated in the Introduction, we focus on the setting wherein the dynamics ($(P^{(S)}, R^{(S)})$ and $(P^{(T)}, R^{(T)})$) of the two MDPs between which we transfer knowledge are defined on different observation spaces but share structural similarities. Specifically, we make the assumption that there exists a mapping between the source and target observation spaces such that the transition dynamics under the mapping in the target task share the same transition dynamics as in the source task. We formalize this in Assumption 1:

Assumption 1. *There exists a function $f : \mathcal{O}^{(T)} \to \mathcal{O}^{(S)}$ such that $\forall o_i^{(T)}, o_j^{(T)} \in \mathcal{O}^{(T)}, \forall a \in \mathcal{A}$,*

$$P^{(T)}(o_j^{(T)}|o_i^{(T)}, a) = P^{(S)}(f(o_j^{(T)})|f(o_i^{(T)}), a), \quad R^{(T)}(o_i^{(T)}, a) = R^{(S)}(f(o_i^{(T)}), a).$$

Remarks. (1) Assumption 1 is mild as many real-world scenarios fall under this assumption. For instance, when upgrading the cameras of a patrol robot to have higher resolutions, such a mapping f can be a down-sampling function. (2) Our algorithm does not assume any prior knowledge of the mapping f, although Assumption 1 assumes the existence of such an underlying mapping. (3) f is a general function without extra restrictions. f can be a many-to-one mapping, i.e., more than one target observations can be related to the same observation in the source task. f can be non-surjective, i.e., there could exist source observations that do not correspond to any target observation.

Many prior works (Mann & Choe, 2013; Brys et al., 2015) have similar assumptions, but require prior knowledge of such an inter-task mapping to achieve knowledge transfer. However, such a mapping might not be available in practice. As an alternative, in the next section, we propose to learn a latent representation of the observations and a dynamics model in this latent space. The dynamics model is transferred to speed up learning in the target task, where we *do not* learn the mapping f explicitly.

4 Methodology: Transfer with Regularized Representation

The main idea of our proposed transfer method is to extract dynamics information on the representations learned in the source task, and utilize the dynamics information to regularize representation learning in the target task. First, we formally characterize the properties of representation learning for RL in Section 4.1, which is crucial to motivate our algorithm. Then, we introduce our proposed algorithm for across-observation transfer in Section 4.2. Theoretical analysis for the proposed algorithm is provided in Section 4.3. The analysis in Section 4.1 and 4.3 mainly focus on learning with policy iteration, and we provide analysis for value-based learning in Appendix C.

4.1 Characterizing Conditions for Good Representations

As discussed in Section 2, real-world applications usually have rich and redundant observations, where learning a good representation (Jaderberg et al., 2016; Dabney et al., 2020) is essential for

efficiently finding an optimal policy. However, the properties that constitute a good representation for an RL task is still an open question. In this section, we formulate conditions for a good representation and the relation between representation and learning performance. Some prior works (Bellemare et al., 2019; Dabney et al., 2020; Gelada et al., 2019) also discuss the representation quality in DRL, but we focus on (1) formulating sufficient conditions for a representation to enable guaranteed learning, and (2) achieving representation sufficiency with transferable dynamics models.

Given a representation mapping ϕ, the Q value of any $(o, a) \in \mathcal{O} \times \mathcal{A}$ can be approximately represented by a function of $\phi(o)$, i.e., $\hat{Q}(o, a) = f(\phi(o); \theta_a)$, where θ_a is a set of learnable parameters. For example, if linear approximation is used, the Q value can be represented as $\hat{Q}(o, a) = \phi(o)\theta_a$, where $\theta_a \in \mathbb{R}^d$ is a vector associated with action a. To study the relation between representation quality and approximation quality, we define an *approximation operator* $\mathcal{H}_\phi$ of the Q value such that $\mathcal{H}_\phi Q(o) := h(\phi(o); \Theta^*)$, where $\Theta^* = \text{argmin}_\Theta \mathbb{E}[\|h(\phi(o); \Theta) - Q(\phi, \cdot)\|]$. Such an approximation can be realized by neural networks as universal function approximators (Hornik et al., 1989). That is, the approximation error $\|Q - \hat{Q}\|$ only depends on the selection of the representation ϕ.

The quality of the encoder ϕ is crucial for learning an accurate value function or learning a good policy. The ideal encoder ϕ should discard irrelevant information in the raw observation but keep essential information. In supervised or self-supervised representation learning (Chen et al., 2020; Achille & Soatto, 2018), it is believed that a good representation Z^* of input X should be (1) *sufficient*, i.e., same mutual information $I(Z^* \wedge Y) = I(X \wedge Y)$ for the underlying label Y, and (2) *minimal*, i.e., smallest mutual information $I(Z^* \wedge X) \leq I(\tilde{Z} \wedge X)$ for all sufficient $\tilde{Z}$. However, in RL, it is difficult to identify whether a representation is sufficient, since there is no label corresponding to each input. The focus of an agent is to estimate the value of each input $o \in \mathcal{O}$, which is associated with some policy. Therefore, we point out that the representation quality in RL is *policy-dependent*. Below, we formally characterize the sufficiency of a representation mapping in terms of a fixed policy and learning a task.

If the agent is executing a fixed policy, and its goal is to estimate the expected future return from the environment, then a representation is sufficient for the policy as long as it can encode the policy value V_π. A formal definition is provided by Definition 9 in Appendix A.

However, the goal of RL is to *find* an optimal policy. Therefore, it is important to investigate the conditions under which a representation is sufficient for a changing policy distribution. Intuitively, a representation mapping is sufficient for learning if we are able to find an optimal policy over the representation space $\phi(\mathcal{O})$, which requires multiple iterations of policy evaluation and policy improvement. Definition 2 below defines a set of "important" policies for learning with $\phi(\mathcal{O})$.

Definition 2 (Encoded Deterministic Policies)**.** *For a given representation mapping $\phi(\cdot)$, define an encoded deterministic policy set Π_ϕ^D as the set of policies that are deterministic and take the same actions for observations with the same representations. Formally,*

$$\Pi_\phi^D := \{\pi \in \Pi : \exists \tilde{\pi} : \phi(\mathcal{O}) \to \mathcal{A},\ \textit{s.t.}\ \forall o \in \mathcal{O}, \pi(o) = \tilde{\pi}(\phi(o))\}.$$

A policy π is in Π_ϕ^D if it does not distinguish o_1 and o_2 when $\phi(o_1) = \phi(o_2)$. Therefore, Π_ϕ^D can be regarded as deterministic policies that make decisions for encoded observations. Now, we define the concept of sufficient representation for learning in an MDP.

Definition 3 (Sufficient Representation for Learning)**.** *A representation mapping ϕ is **sufficient** for a task M w.r.t. approximation operator $\mathcal{H}_\phi$ if $\mathcal{H}_\phi Q_\pi = Q_\pi$, for all $\pi \in \Pi_\phi^D$. (ϕ is **ϵ-sufficient** for learning if $\|\mathcal{H}_\phi Q_\pi - Q_\pi\| \leq \epsilon$ and **linearly-sufficient** if $\mathcal{H}_\phi$ is linear projection.)*

Definition 3 suggests that the representation is sufficient for learning a task as long as it is sufficient for policies in Π_ϕ^D. Then, the lemma below justifies that a nearly sufficient representation can ensure that approximate policy iteration converges to a near-optimal solution.

Lemma 4 (Error Bound for Approximate Policy Iteration)**.** *If ϕ is ϵ-sufficient for task M (with ℓ_∞ norm), then the approximated policy iteration with approximation operator $\mathcal{H}_\phi$ starting from any initial policy that is encoded by ϕ ($\pi_0 \in \Pi_\phi^D$) satisfies*

$$\limsup_{k \to \infty} \|Q^* - Q^{\pi_k}\|_\infty \leq \frac{2\gamma^2 \epsilon}{(1-\gamma)^2}, \tag{2}$$

where π_k is the policy in the k-th iteration.

Lemma 4, proved in Appendix B, is extended from the error bound provided by Bertsekas & Tsitsiklis (1996). For simplicity, we consider the bound in ℓ_∞, but tighter bounds can be derived with other norms (Munos, 2005), although a tighter bound is not the focus of this paper.

How Can We Learn A Sufficient Representation? So far we have provided a principle to justify whether a given representation is sufficient for learning. In DRL, the representation is learned together with the policy or value function using neural networks, but the quality of the representation may be poor (Dabney et al., 2020), which makes it hard for the agent to find an optimal policy. Based on Definition 3, a natural method to learn a good representation is to let the representation fit as many policy values as possible as auxiliary tasks, which matches the ideas in other works. For example, Bellemare et al. (2019) propose to fit a set of representative policies (called *adversarial value functions*). Dabney et al. (2020) choose to fit the values of all past policies (along the *value improvement path*), which requires less computational resource.

Different from these works that directly fit the value functions of multiple policies, we propose a computationally efficient way to realize sufficient representation for learning and knowledge transferring: *to fit an auxiliary policy-independent dynamics model.* In Section 4.2, we introduce our algorithm that uses a transferable model-based regularizer to facilitate learning. Then, in Section 4.3, we provide detailed theoretical analysis for why the dynamics model helps to learn a good representation and why it is transferable.

4.2 Algorithm: Learning and Transferring Model-based Regularizer

Our goal is to use the knowledge learned in the source task to learn a good representation in the target task, such that the agent learns the target task more easily than learning from scratch. Since we focus on developing a generic transfer mechanism, the base learner can be any DRL algorithms. We use L_{base} to denote the loss function of the base learner.

As motivated in Section 4.1, we propose to learn policy-independent dynamics models for producing high-quality representations: (1) $\hat{P}$ which predicts the representation of the next state for a given state action pair, and (2) $\hat{R}$ which predicts the immediate reward for a given state-action pair. For a batch of N transition samples $\{o_i, a_i, o_i', r_i\}_{i=1}^N$, define the transition loss and the reward loss as:

$$L_P(\phi, \hat{P}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{P}(\phi(o_i), a_i) - \bar{\phi}(o_i'))^2, \quad L_R(\phi, \hat{R}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{R}(\phi(o_i), a_i) - r_i)^2 \tag{3}$$

where $\bar{\phi}(o_i')$ denotes the representation of the next state o_i' with stop gradients.

The learning procedures for the source task and the target task are illustrated in Algorithm 1 and Algorithm 2, respectively. Figure 2 depicts the architecture of the learning model for both source and target tasks. $z = \phi(o)$ and $z' = \bar{\phi}(o')$ are the encoded observation and next observation. Given the current encoding z and the action a, the dynamics models $\hat{P}$ and $\hat{R}$ return the predicted next encoding $\hat{z}' = \hat{P}(z, a)$ and predicted reward $\hat{r} = \hat{R}(z, a)$. Then the transition loss is the mean squared error (MSE) between z' and $\hat{z}'$ in a batch; the reward loss is the MSE between r and $\hat{r}$ in a batch.

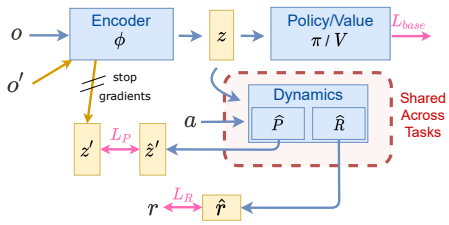


Figure 2: The architecture of proposed method. $\hat{P}$ and $\hat{R}$ are learned in the source task, then transferred to the target task and fixed during training.

In the source task (Algorithm 1): dynamics models $\hat{P}$ and $\hat{R}$ are learned by minimizing L_P and L_R, which are computed based on a target encoder $\hat{\phi}^{(S)}$ (Line 5). The computation of a target encoder is to make the learning of dynamics models more stable. Note that a target network is widely used in base DRL algorithms (e.g. DQN), which can be directly used as $\hat{\phi}^{(S)}$ without maintaining an extra network. The encoder $\phi^{(S)}$ is regularized by the auxiliary dynamics models $\hat{P}$ and $\hat{R}$ (Line 6).

In the target task (Algorithm 2): dynamics model $\hat{P}$ and $\hat{R}$ are transferred from the source task and fixed during learning. Therefore, the learning of $\phi^{(T)}$ is regularized by static dynamics models, which leads to faster and more stable convergence than naively learning an auxiliary task.

Algorithm 1 Source Task Learning

Require: Regularization weight λ; update frequency m for target encoder.

1: Initialize encoder $\phi^{(S)}$, target encoder $\hat{\phi}^{(S)}$, policy $\pi^{(S)}$, transition prediction network $\hat{P}$ and reward prediction network $\hat{R}$.
2: **for** $t = 0, 1, \cdots$ **do**
3: Take action $a_t \sim \pi^{(S)}(\phi^{(S)}(o_t^{(S)}))$, get next observation $o_{t+1}^{(S)}$ and reward r_t, store to buffer.
4: Sample a mini-batch $\{o_i, a_i, r_i, o_i'\}_{i=1}^N$ from the buffer.
5: Update $\hat{P}$ and $\hat{R}$ using one-step gradient descent with $\nabla_{\hat{P}} L_P(\hat{\phi}^{(S)}; \hat{P})$ and $\nabla_{\hat{R}} L_R(\hat{\phi}^{(S)}; \hat{R})$, where L_P and L_R are defined in Equation (3).
6: Update encoder and policy by $\min_{\pi^{(S)}, \phi^{(S)}} L_{\text{base}}(\phi^{(S)}, \pi^{(S)}) + \lambda\big(L_P(\phi^{(S)}; \hat{P}) + L_R(\phi^{(S)}; \hat{R})\big)$.
7: **if** $t \mid m$ **then** Update the target encoder $\hat{\phi}^{(S)} \leftarrow \phi^{(S)}$.

Algorithm 2 Target Task Learning with Transferred Dynamics Models

Require: Regularization weight λ; dynamics models $\hat{P}$ and $\hat{R}$ learned in the source task.

1: Initialize encoder $\phi^{(T)}$, policy $\pi^{(T)}$
2: **for** $t = 0, 1, \cdots$ **do**
3: Take action $a_t \sim \pi^{(T)}(\phi^{(T)}(o_t^{(T)}))$, get next observation $o_{t+1}^{(T)}$ and reward r_t, store to buffer.
4: Sample a mini-batch $\{o_i, a_i, r_i, o_i'\}_{i=1}^N$ from the buffer.
5: Update encoder and policy by $\min_{\phi^{(T)}, \pi^{(T)}} L_{\text{base}}(\phi^{(T)}, \pi^{(T)}) + \lambda\big(L_P(\phi^{(T)}; \hat{P}) + L_R(\phi^{(T)}; \hat{R})\big)$.

Network Complexity and Representation Sufficiency It is commonly believed that a good representation should linearly represent the policy values. That is, the policy/value head should be a single linear layer. However, as we discussed in Section 4.1, being linearly sufficient for learning requires the representation to be sufficient for exponentially many policies, which is expensive to compute and hard to transfer. Instead, we encourage the representation to be sufficient for predicting the next state representation and the reward, which is relatively easy to achieve and transfer. As a result, the regularized representation is sufficient but not linearly sufficient for learning. Therefore, we suggest using a *non-linear policy/value head* following the representation. In Section 4.3, we elaborate the relationship between model sufficiency and learning sufficiency of representations, where we show a trade-off between model complexity and approximation complexity.

4.3 Theoretical Analysis

In our algorithm, we let the encoder ϕ minimize L_P and L_R as auxiliary tasks. In this section, we analyze the theoretical insights behind this model-based auxiliary task. By associating model sufficiency and learning sufficiency, we show theoretical justifications and practical advantages of our proposed method in terms of both single-task learning and transfer learning.

For notational simplicity, let P_a and R_a denote the transition and reward functions associated with action $a \in \mathcal{A}$. Note that P_a and R_a are independent of any policy. We then define the sufficiency of a representation mapping w.r.t. dynamics models as below.

Definition 5 (Policy-independent Model Sufficiency)**.** *For an MDP M, a representation mapping ϕ is sufficient for its dynamics $(P_a, R_a)_{a\in\mathcal{A}}$ if $\forall a \in \mathcal{A}$, there exists functions $\hat{P}_a : \mathbb{R}^d \to \mathbb{R}^d$ and $\hat{R}_a : \mathbb{R}^d \to \mathbb{R}$ such that $\forall o \in \mathcal{O}$, $\hat{P}_a(\phi(o)) = \mathbb{E}_{o'\sim P_a(o)}[\phi(o')]$, $\hat{R}_a(\phi(o)) = R_a(o)$.*

Remarks. (1) Since we focus on learning a deterministic dynamics model $\hat{P}$, ϕ is exactly sufficient for dynamics $(P_a, R_a)_{a\in\mathcal{A}}$ only when the transition function P is deterministic. (2) If P is stochastic, but we have $\max_{o,a} \|\mathbb{E}_{o'\sim P_a(o)}[\phi(o')] - \hat{P}_a(\phi(o))\| \le \epsilon_P$ and $\max_{o,a} |R_a(o) - \hat{R}_a(\phi(o))| \le \epsilon_R$, then ϕ is **(ϵ_P, ϵ_R)-sufficient** for the dynamics of M.

Next we show by Proposition 6 and Theorem 7 that learning sufficiency can be achieved via ensuring model sufficiency.

Proposition 6 (Learning Sufficiency Induced by Policy-independent Model Sufficiency)**.** *Consider an MDP M with deterministic transition function P and reward function R. If ϕ is sufficient for $(P_a, R_a)_{a\in\mathcal{A}}$, then it is sufficient for learning in M.*

Proposition 6 shows that, if the transition is deterministic and the model errors L_P, L_R are zero, then ϕ is exactly sufficient for learning. More generally, if the transition function P is not deterministic, and model fitting is not perfect, the learned representation can still be nearly sufficient for learning as characterized by Theorem 7 below, which is extended from a variant of the value difference bound derived by Gelada et al. (2019). These results justify that learning the latent dynamics models as an auxiliary task helps with the representation quality.

Theorem 7. *For an MDP M, if representation mapping ϕ is (ϵ_P, ϵ_R)-sufficient for the dynamics of M, then approximate policy iteration with approximation operator $\mathcal{H}_\phi$ starting from any initial policy $\pi_0 \in \Pi_\phi^D$ satisfies*

$$\limsup_{k\to\infty} \|Q^* - Q^{\pi_k}\|_\infty \leq \frac{2\gamma^2}{(1-\gamma)^3}(\epsilon_R + \gamma\epsilon_P K_{\phi,V}). \tag{4}$$

where $K_{\phi,V}$ is an upper bound of the value Lipschitz constant as defined in Appendix A.

Transferring Dynamics to Get Better Representation in Target. Although Proposition 6 shows that learning policy-independent dynamics models can benefit representation learning, finding the optimal solution is not trivial since $\hat{P}_a$ and $\hat{R}_a$ are unknown and have to be learned. Therefore, the main idea of our algorithm is to transfer the dynamics models $\hat{P}_a, \hat{R}_a$ from the source task to the target task, to significantly ease the learning in the target task. Theorem 8 below guarantees that transferring the dynamics models is feasible. Our experimental result in Section 6 verifies that learning with transferred dynamics models outperforms learning with auxiliary dynamics models.

Theorem 8 (Transferable Dynamics Models)**.** *Consider a source task $M^{(S)}$ and a target task $M^{(T)}$ with deterministic transition functions. Suppose $\phi^{(S)}$ is sufficient for $(P_a^{(S)}, R_a^{(S)})_{a\in\mathcal{A}}$ with functions $\hat{P}_a, \hat{R}_a$, then there exists a representation $\phi^{(T)}$ satisfying $\hat{P}_a(\phi(o)) = \mathbb{E}_{o'\sim P_a^{(T)}}[\phi(o')]$, $\hat{R}_a(\phi(o)) = R_a^{(T)}(o)$, for all $o \in \mathcal{O}^{(T)}$, and $\phi^{(T)}$ is sufficient for learning in $M^{(T)}$.*

Theorem 8 shows that the learned latent dynamics models $\hat{P}_a, \hat{R}_a$ are transferable from the source task to the target task. For simplicity, Theorem 8 focuses on exact sufficiency as in Proposition 6, but it can be easily extended to ϵ-sufficiency if combined with Theorem 7. Proofs for Proposition 6, Theorem 7 and Theorem 8 are all provided in Appendix B.

Why Non-linear Policy Head Fitting policy-independent dynamics encourages the representation to be sufficient for learning, but not linearly sufficient. Therefore, we set the policy head after the latent representation to be a non-linear network to reduce approximation error. Linear sufficiency can be achieved if ϕ is made linearly sufficient for P_π and R_π for all π, where P_π and R_π are transition and reward functions induced by policy π (Proposition 10, Appendix A). However, using this method for transfer learning is expensive in terms of both computation and memory, as it requires to learn P_π and R_π for many different π's and store these models for transferring to the target task. Therefore, there is a trade-off between approximation complexity and representation complexity. Learning a linearly sufficient representation reduces the complexity of the approximation operator. But it requires more complexity in the representation itself as it has to satisfy much more constraints. To develop a practical and efficient transfer method, we use a slightly more complex approximation operator (non-linear policy head) while keeping the auxiliary task simple and easy to transfer across tasks. Please see Appendix A for more detailed discussion about linear sufficiency.

5 Related Work

Transfer RL across Observation Feature Spaces Transferring knowledge between tasks with different observation spaces has been studied for years. Many existing approaches(Taylor et al., 2007; Mann & Choe, 2013; Brys et al., 2015) require an explicit mapping between the source and target observation spaces, which may be hard to obtain in practice. Sun et al. (2020) propose a provably sample efficient transfer learning algorithm that works for different observation spaces without knowing any inter-task mapping, but the algorithm is designed for tabular RL and not easy to scale up to the DRL regime. Raiman et al. (2019) introduce network surgery that deals with the change in the input features by determining which components of a neural network model should be transferred from a source model and which require retraining. However, when the change is drastic, e.g. vector to pixel, transfer will not happen. Gupta et al. (2017) achieve transfer learning between two different tasks by learning an invariant feature space, but a key assumption is that the source and the target domain will encounter the same latent state at the same timestep, which does not hold

when the environments are stochastic, or when the agent only has access to one environment. A similar topic is domain adaptation in RL (Higgins et al., 2017; Eysenbach et al., 2020; Zhang et al., 2020), where the target observation space (e.g. real world) is different from the source observation (e.g. simulator). However, domain adaptation is different from our setting as the source and target observations still have the same shapes, although having different demonstrations, e.g. the noise in background. The aim of domain adaptation is to train a policy in the source domain that can generalize to the target domain, thus prior knowledge or a few samples of the target domain is often needed (Eysenbach et al., 2020). Our work, in contrast, transfers knowledge between observations with different shapes, and aims for few-shot learning in the target task.

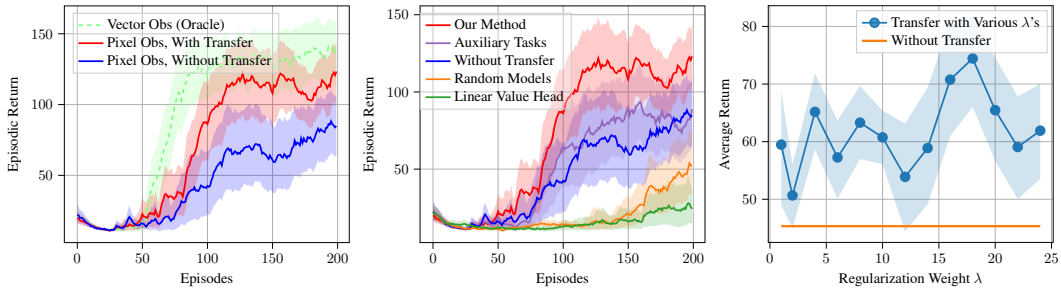
Representation Learning in RL In environments with rich observations, representation learning is crucial for the efficiency of RL methods. Learning unsupervised auxiliary tasks (Jaderberg et al., 2016) is shown to be effective for learning a good representation. The relationship between learning policy-dependent auxiliary tasks and learning good representations has been studied in some prior works (Bellemare et al., 2019; Dabney et al., 2020; Lyle et al., 2021), while our focus is to learn policy-independent auxiliary tasks to facilitate transfer learning. Using latent prediction models to regularize representation has been shown to be effective for various types of rich observations (Guo et al., 2020; Lee et al., 2019). Gelada et al. (2019) theoretically justify that learning latent dynamics model guarantees the quality of the learned representation, while we further characterize the relationship between representation and learning performance, and we utilize dynamics models to improve transfer learning. Zhang et al. (2020) use a bisimulation metric to learn latent representations that are invariant to task-irrelevant details in observation. As pointed out by Achille & Soatto (2018), invariant and sufficient representation is indeed minimal sufficient, so it is an interesting future direction to combine our method with bisimulation metric to learn minimal sufficient representations. There is also a line of work using contrastive learning to train an encoder for pixel observations (Srinivas et al., 2020; Yarats et al., 2021; Stooke et al., 2021), which usually pre-train an encoder based on image samples using self-supervised learning. However, environment dynamics are usually not considered during pre-training. Our algorithm can be combined with these contrastive learning approaches to further improve learning performance in the target task.

6 Experimental Evaluation

In this section, we empirically evaluate our transfer learning algorithm in various environments and multiple observation changes. The detailed setup and hyperparameters are contained in Appendix D. As this paper studies a novel problem, we tried but could not find any existing method that works under the same setting. Therefore, we evaluate our algorithm by (1) comparing it with the base learning algorithm, and (2) performing ablation studies for different components of the algorithm.

Transfer Knowledge from Vector Observations to Pixel Observations To verify that our transfer algorithm works for drastic observation changes, we first focus on a challenging scenario where we transfer knowledge from a compact observation space with vector features to a rich observation space with pixel images. We implement a DQN learner (Mnih et al., 2015) on the classic CartPole environment in Gym (Brockman et al., 2016). The source-task agent observes a 4-dimensional vector containing positions and velocities of cart and pole, while in the target task, the agent observes a screenshot of the rendered environment. Figure 3a shows a comparison between learning with our transfer algorithm and learning without our transfer algorithm, where the source task learning curve serves as an oracle. We can see that with our proposed transfer method, the agent learns the pixel observation much more efficiently than a single-task learning agent in the challenging target task with pixel observations, and even impressively achieves similar performance as in the simpler vector-based source task. *To the best of our knowledge, we are the first to achieve knowledge transfer from a vector-input environment to a pixel-input environment without any pre-defined mappings.*

Ablation Study and Hyper-parameter Test We conduct an ablation study to evaluate each component of our algorithm as shown in Figure 3b. We find that when transferring the dynamics models with only a linear value head (the **green** curve), the agent fails to learn a good policy as we analyzed in Section 4.3. If the dynamics models $(\hat{P}, \hat{R})$ are randomly generated instead of being transferred from the source task (the **orange** curve), the agent does not learn, either. More importantly, if we learn dynamics models as auxiliary tasks in the target task without transferring them from the source (the **purple** curve), the agent learns a little better than a vanilla agent, but is worse than our proposed transfer algorithm. These empirical results have verified our theoretical insights and shown



(a) Transfer to pixel CartPole. **(b)** Ablation study **(c)** Hyperparameter test

Figure 3: (a) By transferring knowledge from vector-input CartPole to pixel-input CartPole, the agent achieves much better performance than learning the pixel-input CartPole from scratch. (b) Ablation study verifies the effectiveness of our algorithm design. (c) Under different selections of hyperparameter λ, the algorithm works better than learning from scratch (when $\lambda = 0$). All results are averaged over 20 random seeds.

the effectiveness of our algorithm design. Figure 3c further visualizes how the hyperparameter λ (regularization weight) influences the transfer performance. It can be found that the agent generally benefits from a larger λ, which suggests that the model-based regularization has a positive impact on the learning performance. For a wide range of λ's, the agent always outperforms the learner without transfer (the learner with $\lambda = 0$). Therefore, our algorithm is not sensitive to the hyperparameter λ, and a larger λ is preferred to get better performance.

Transfer Knowledge for Harder Continuous Control We further evaluate our transfer learning algorithm in multiple continuous control tasks. HalfCheetah, Hopper and Walker2d are classic MuJoCo environments, where we increase the number of input features in the target tasks (See Appendix D for details). 3DBall is a ball-balancing game contained in the Unity ML-Agents Toolkit (Juliani et al., 2018), which has two different observation specifications that naturally fit our transfer setting: the source observation has 8 features containing the velocity of the ball; the target observation (harder version) does not have the ball's velocity, but stacks the past 9 frames. The results are shown in Figure 4, where our transfer algorithm always outperforms the vanilla learner without knowledge transfer. An ablation study is provided in Figure 5 in Appendix D, where we show that transferring only $\hat{P}$ or transferring only $\hat{R}$ are both better than a single-task learner, but worse than our method that transfers both $\hat{P}$ and $\hat{R}$.

In some environments such as HalfCheetah, our transfer algorithm significantly outperforms the baseline without transfer. But in Walker2d, the improvement is relatively small, although transferring is still better than not transferring. This phenomenon is common in model-based learning (Nagabandi et al., 2018), as state predicting in Walker2d is harder than HalfCheetah due to the complexity of the dynamics. Therefore, we suggest using our method to transfer when the learned models $(\hat{P}, \hat{R})$ in the source task are relatively good (have low error). More techniques of improving model-based learning, such as bisimulation (Zhang et al., 2020), can be applied to further improve the transfer performance.

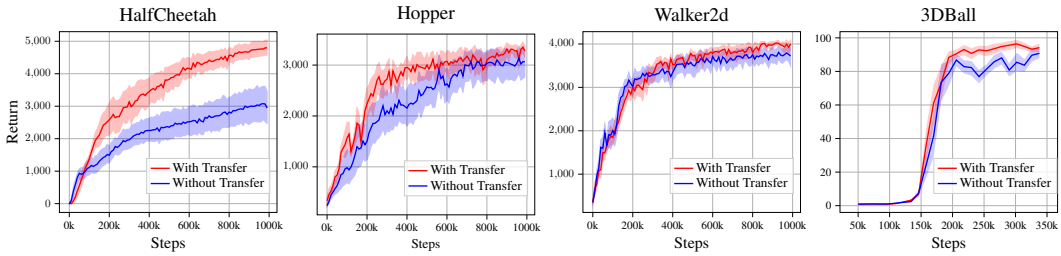


Figure 4: Comparison of single-task SAC and SAC combined with our transfer method in the target task. All results are averaged over 10 random seeds.

7 Conclusion

In this paper, we identify and propose a solution to an important but rarely studied problem: transferring knowledge between tasks with drastically different observation spaces where inter-task mappings are not available. We propose to learn a latent dynamics model in the source task and transfer the model to the target task to facilitate representation learning. Theoretical analysis and empirical study justify the effectiveness of the proposed algorithm.

ETHICS STATEMENT

Transfer learning aims to apply previously learned experience to new tasks to improve learning efficiency, which is becoming more and more important nowadays for training intelligent agents in complex systems. This paper focuses on a practical but rarely studied transfer learning scenario, where the observation feature space of an RL environment is subject to drastic changes. Driven by theoretical analysis on representation learning and its relation to latent dynamics learning, we propose a novel algorithm that transfers knowledge between tasks with totally different observation spaces, without any prior knowledge of an inter-task mapping.

This work can benefit many applications as suggested by the examples below.
(1) In many real-life environments where deep RL is used (e.g. navigating in a building), the underlying dynamics (e.g. the structure of the building) are usually fixed, but what features the observation space has is designed by human developers (e.g. what sensors are installed) and thus may change frequently during the development. When the agent gets equipped with better sensors, our algorithm makes it possible to reuse previously learned models when learning with the new sensors.
(2) An agent usually learns better with a compact observation space (e.g. a low-dimensional vector space containing its location and the goal's location) than a rich/noisy observation space (e.g. an image containing the goal). However, a compact observation is usually more difficult to construct in practice as it may require expert knowledge and human work. In this case, one can extract compact observations in a few samples and pre-train a policy with our Algorithm 1, then train the agent in the real environment with rich observations with our Algorithm 2 using the learned dynamics models. Our experiment in Figure 3 shows that the learning efficiency in the rich-observation environment can be significantly improved with our proposed transfer method.

REPRODUCIBILITY STATEMENT

For theoretical results, we provide concrete proofs in Appendix B and Appendix C. For empirical results, we illustrate implementation details in Appendix D. The source code and running instructions are provided in the supplementary materials.

REFERENCES

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research*, 19(1):1947–1980, 2018.

Marc Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 32:4358–4369, 2019.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-dynamic programming.* Athena Scientific, Belmont, MA, 1996.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Tim Brys, Anna Harutyunyan, Matthew E Taylor, and Ann Nowé. Policy transfer using reward shaping. In *AAMAS*, pp. 181–188, 2015.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G. Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. *CoRR*, abs/2006.02243, 2020. URL https://arxiv.org/abs/2006.02243.

Benjamin Eysenbach, Swapnil Asawa, Shreyas Chaudhari, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*, 2020.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pp. 2170–2179. PMLR, 2019.

Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, pp. 3875–3886. PMLR, 2020.

Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pp. 1480–1490. PMLR, 2017.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

Ronald A Howard. Dynamic programming and markov processes. 1960.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.

Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *CoRR*, abs/1907.00953, 2019. URL http://arxiv.org/abs/1907.00953.

Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *International Conference on Artificial Intelligence and Statistics*, pp. 1–9. PMLR, 2021.

Timothy A Mann and Yoonsuck Choe. Directed exploration in reinforcement learning with transferred knowledge. In *European Workshop on Reinforcement Learning*, pp. 59–76. PMLR, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Rémi Munos. Error bounds for approximate value iteration. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, pp. 1006. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566, 2018. doi: 10.1109/ICRA.2018.8463189.

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

Jonathan Raiman, Susan Zhang, and Christy Dennison. Neural network surgery with sets. *arXiv preprint arXiv:1912.06719*, 2019.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pp. 9870–9879. PMLR, 2021.

Yanchao Sun, Xiangyu Yin, and Furong Huang. Temple: Learning template of transitions for sample efficient multi-task rl. *arXiv preprint arXiv:2002.06659*, 2020.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

Matthew E Taylor, Peter Stone, and Yaxin Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(9), 2007.

Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. *arXiv preprint arXiv:2102.11271*, 2021.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020.

Appendix: Transfer RL across Observation Feature Spaces via Model-Based Regularization

A Additional Discussion for Representation Sufficiency

Good Representation for A Fixed Policy The following definition characterizes the sufficiency of a representation mapping in terms of evaluating a fixed policy.

Definition 9 (Sufficient Representation for A Fixed Policy)**.** *A representation mapping ϕ is **sufficient** for a policy π w.r.t. approximation operator $\mathcal{H}_\phi$ iff $\mathcal{H}_\phi V^\pi = V^\pi$. More generally, for a constant $\epsilon \geq 0$, Φ is **ϵ-sufficient** for π iff $\|\mathcal{H}_\phi V^\pi - V^\pi\| \leq \epsilon$.*

Remarks. (1) If $o_1, o_2 \in \mathcal{O}$ have different values under π, a good representation should be able to distinguish them, i.e., $\phi^*(o_1) \neq \phi^*(o_2)$.
(2) The approximation operator $\mathcal{H}_\phi$ is linear if $\mathcal{H}_\phi V = \text{Proj}_\Phi(V)$ where $\text{Proj}_\Phi(\cdot)$ denotes the orthogonal projection to the subspace spanned by the basis functions of $\langle \phi_1, \phi_2, \cdots, \phi_d \rangle$.

Model Sufficiency over Policies Induces Linear Learning Sufficiency It can be found from Definition 3 that ϕ is sufficient as long as it represents Q_π for all $\pi \in \Pi_\phi^D$. Fitting various value functions to improve representation quality is proposed by some prior works Bellemare et al. (2019); Dabney et al. (2020) and shown to be effective. However, learning and fitting many policy values could be computationally expensive, and is not easy to be applied to transfer learning between tasks with different observation spaces. Can we regularize the representation with the latent dynamics instead of policy values? Proposition 10 below shows that if Φ is linearly sufficient for all dynamics pairs (P_π, R_π) induced by policies in Π_ϕ^D and the dynamics pairs associated with all actions, then Φ is linearly sufficient for learning.

For notation simplicity, assume the state space is finite. Then, let (P_π, R_π) be the transition matrix and reward vector induced by policy π, i.e., $P_\pi[i,j] = \mathbb{E}_{a\sim\pi(o_i)}[P(o_j|o_i,a)]$, $R_\pi[i] = \mathbb{E}_{a\sim\pi(o_i)}[R(o_i,a)]$. Similarly, let $P_a[i,j] = P(o_j|o_i,a)$, $R_a[i] = R(o_i,a)$. We let Φ denote the representation matrix, where the i-th row of Φ refers to the feature of the i-the observation.

Proposition 10 (Linear Sufficiency Induced by Policy-based Model Sufficiency)**.** *For representation Φ, if for all $\pi \in \Pi_\phi^D, a \in \mathcal{A}$, there exist $\hat{P}_\pi, \hat{R}_\pi, \hat{P}_a, \hat{R}_a$ such that $\Phi\hat{P}_\pi = P_\pi\Phi$, $\Phi\hat{R}_\pi = R_\pi$, $\Phi\hat{P}_a = P_a\Phi, \Phi\hat{R}_a = R_a$, i.e. Φ is linearly sufficient both policy-based dynamics and policy-independent dynamics models, i.e., then Φ is linearly sufficient for a task M w.r.t. approximation operator $\mathcal{H}_\phi$.*

Proposition 10 proven in Appendix B.3 suggests that we can let representation fit (P_π, R_π) for many different π's. However, it could be computationally intractable since the policy space is large. More importantly, it is hard to store and transfer a large number of dynamics models for all $(\hat{P}_\pi, \hat{R}_\pi)$. In our Proposition 6, we show that learning sufficiency can be induced by policy-independent model sufficiency, which is much simpler as there is no need to learn and store $(\hat{P}_\pi, \hat{R}_\pi)$ for many policies. As a trade-off, the policy-independent model induces non-linear sufficiency instead of linear sufficiency, requiring a more expressive approximation operator.

Latent MDP induced by Representation We follow the analysis by Gelada et al. (2019) and define a new MDP under the representation mapping ϕ: $\tilde{M} = \langle \tilde{\mathcal{O}}, \mathcal{A}, \tilde{P}, \tilde{R}, \gamma \rangle$, where for all $o \in \mathcal{O}$, $\phi(o) \in \tilde{\mathcal{O}}$, $\tilde{P}(\phi(o), a) = \hat{P}_a\phi(o)$, $\tilde{R}(\phi(o), a) = \hat{R}_a\phi(o)$. Let $\tilde{V}$ denote the value function in $\tilde{M}$, and let $\tilde{\pi}$ denote a policy in $\tilde{M}$. We make the following mild assumption

Assumption 11. *There exists a constant $K_{\phi,V}$ such that*

$$|\tilde{V}_{\tilde{\pi}}(\phi(o_1)) - \tilde{V}_{\tilde{\pi}}(\phi(o_2))| \leq K_{\phi,V}\|\phi(o_1) - \phi(o_2)\|, \forall \tilde{\pi} : \tilde{O} \to \mathcal{A}, o_1, o_2 \in \mathcal{O}.$$

This assumption is mild as any MDP with bounded reward has bounded value functions.

B TECHNICAL PROOFS

B.1 PROOF OF LEMMA 4

Proof of Lemma 4. We first show that policy iteration with approximation operator $\mathcal{H}_\phi$ starting from a policy $\pi_0 \in \Pi_\phi^D$ generates a sequence of policies that are in Π_ϕ^D. That is, for all π_k, and any $o_1, o_2 \in \mathcal{O}$, $\pi_k(o_1) = \pi_k(o_2)$ if $\phi(o_1) = \phi(o_2)$. We prove this claim by induction.

Base case: when $k = 0$, $\pi_0 \in \Pi_\phi^D$.

Inductive step: assume $\pi_k \in \Pi_\phi^D$ for $k \geq 0$, then for iteration $k+1$, we know that

$$\pi_{k+1}(o) := \text{argmax}_a Q_k(o, a) \tag{5}$$

where $Q_k = \mathcal{H}_\phi Q_{\pi_k}$.

Based on the definition of $\mathcal{H}_\phi$, $Q_k(o, a) = f(\phi(o); \theta_a)$ for some θ_a. Hence, if o_1 and o_2 have the same representation, $Q_k(o_1, \cdot)$ and $Q_k(o_2, \cdot)$ are equal. As a result, $\pi_{k+1}(o_1)$ and $\pi_{k+1}(o_2)$ are equal, so $\pi_{k+1} \in \Pi_\phi^D$.

Next we prove the error bound in Lemma 4. We start by restating the error bounds for approximate policy iteration by Bertsekas & Tsitsiklis (1996):

$$\text{limsup}_{k\to\infty} \|V^* - V^{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \text{sup}_k \|V_k - V^{\pi_k}\|_\infty \tag{6}$$

where π_k is the policy in the k-th iteration. Then we extend the above result to the action value Q.

For any π_k during the policy iteration (as proven above, $\pi_k \in \Pi_\phi^D$), if ϕ is ϵ-sufficient for M as defined in Definition 3 with ℓ_∞ norm, then we have $\|Q_k - Q_{\pi_k}\|_\infty \leq \epsilon$. That is, $\forall o \in \mathcal{O}, a \in \mathcal{A}, |Q_k(o, a) - Q_{\pi_k}(o, a)| \leq \epsilon$. Therefore, $\forall o \in \mathcal{O}$,

$$|V_k(o) - V_{\pi_k}(o)| = |\sum_{a\in\mathcal{A}} \pi(a|o)(Q_k(o, a) - Q_{\pi_k}(o, a))| \leq \epsilon \tag{7}$$

On the other hand, we can derive

$$\|Q^* - Q_{\pi_k}\|_\infty = \max_{o,a} |Q^*(o, a) - Q_{\pi_k}(o, a)| \tag{8}$$

$$= \max_{o,a} |R(o, a) + \gamma \sum_{o'\in\mathcal{O}} P(o'|o, a)V^*(o') - R(o, a) - \gamma \sum_{o'\in\mathcal{O}} P(o'|o, a)V_{\pi_k}(o')| \tag{9}$$

$$= \gamma \|V^* - V_{\pi_k}\|_\infty \tag{10}$$

Combining Equation (7) and (10), we obtain

$$\text{limsup}_{k\to\infty} \|Q^* - Q_{\pi_k}\|_\infty \leq \frac{2\gamma^2}{(1-\gamma)^2} \epsilon. \tag{11}$$

□

B.2 PROOF OF PROPOSITION 6

Proof of Proposition 6. Given that P is deterministic for $o \in \mathcal{O}, a \in \mathcal{A}$, we slightly abuse notation and let $o' = P(o, a) = P_a(o)$ if $P(o'|o, a) = 1$. If ϕ is sufficient for the dynamics models, i.e. $\forall o \in \mathcal{O}, a \in \mathcal{A}$, $\hat{P}_a\phi(o) = \phi(P_a(o))$, $\hat{R}_a\phi(o) = R_a(o)$. Then, we can define a new MDP $\tilde{M} = \langle \tilde{\mathcal{O}}, \mathcal{A}, \tilde{P}, \tilde{R}, \gamma \rangle$, where for all $o \in \mathcal{O}$, $\phi(o) \in \tilde{\mathcal{O}}$, and $\tilde{P}(\phi(o), a) = \hat{P}_a\phi(o) = \phi(P(o, a))$, $\tilde{R}(\phi(o), a) = \hat{R}_a\phi(o) = R(o, a)$.

Any policy $\pi \in \Pi_\phi^D$, based on the definition of Π_ϕ^D, can be written as $\tilde{\pi} \circ \phi$, where $\tilde{\pi}$ is a deterministic policy in $\tilde{M}$. Next, we show that for all $o \in \mathcal{O}, a \in \mathcal{A}$, $Q_\pi(o) = \tilde{Q}_{\tilde{\pi}}(\phi(o))$.

By definition of the Q value, we know

$$Q_\pi(o,a) = \mathbb{E}_{\pi,P}[\sum_{t=0}^{\infty} \gamma^t R(o_t,a_t)|o_0 = o, a_0 = a] \tag{12}$$

$$\tilde{Q}_{\tilde{\pi}}(\phi(o)) = \mathbb{E}_{\tilde{\pi},\tilde{P}}[\sum_{t=0}^{\infty} \gamma^t \tilde{R}(\tilde{o}_t,\tilde{a}_t)|\tilde{o}_0 = \phi(o), \tilde{a}_0 = a] \tag{13}$$

We claim that in the above equations, $\tilde{o_t} = \phi(o_t)$, $\tilde{a_t} = a_t$, for all $t \geq 0$. We prove the claim by induction.

When $t = 0$, the claim holds as $\tilde{o_0} = \phi(o_0) = \phi(o)$, $\tilde{a_0} = a_0 = a$.

Then, with inductive hypothesis that $\tilde{o_t} = \phi(o_t)$, $\tilde{a_t} = a_t$, we show the claim holds for $t+1$:

Action: $a_{t+1} = \pi(o_{t+1}) = \tilde{\pi}(\phi(o_{t+1})) = \tilde{a}_{t+1}$.

State: $\tilde{o}_{t+1} = \tilde{P}(\tilde{o_t}, a_t) = \tilde{P}(\phi(o_t), a_t) = \phi(P(o_t, a_t)) = \phi(o_{t+1})$.

Hence, we have shown $\tilde{o}_t = \phi(o_t)$, $\tilde{a}_t = a_t$, for all $t \geq 0$, then the reward in the t-th step of Equation (12) and (13) are the same, as $\tilde{R}(\tilde{o_t}, \tilde{a_t}) = \tilde{R}(\phi(o_t), a_t) = R(o_t, a_t)$. Therefore, $Q_\pi(o) = \tilde{Q}_{\tilde{\pi}}(\phi(o))$.

Therefore, for any $\pi \in \Pi_\phi^D$, its action value can be represented by $\tilde{Q}_{\tilde{\pi}} \circ \phi$. Therefore, ϕ is sufficient for learning in M.

□

B.3 Proof of Proposition 10

Proof of Proposition 10. We first show that for any $\pi \in \Pi$, if ϕ is linearly sufficient for (P_π, R_π), then there exists a vector $\omega \in \mathbb{R}^k$ such that $V_\pi = \hat{V}_\pi = \Phi\omega$.

Since Φ is linearly sufficient for P_π and R_π, we have $\Phi\hat{P}_\pi = P_\pi\Phi$ and $\Phi\hat{R}_\pi = R_\pi$ for some $\hat{P}_\pi$ and $\hat{R}_\pi$. Let $\omega = (I - \gamma\hat{P}_\pi)^\dagger \hat{R}_\pi$, then the Bellman error of $\hat{V}_\pi = \Phi\omega$ can be computed as

$$\begin{aligned}
R_\pi + \gamma P_\pi \hat{V}_\pi - \hat{V}_\pi &= R_\pi + \gamma P_\pi \Phi\omega - \Phi\omega \\
&= \Phi\hat{R}_\pi + \gamma\Phi\hat{P}_\pi\omega - \Phi\omega \\
&= \Phi(\hat{R}_\pi - (I - \gamma\hat{P}_\pi)(I - \gamma\hat{P}_\pi)^\dagger \hat{R}_\pi) \\
&= \Phi(\hat{R}_\pi - \hat{R}_\pi) \\
&= 0
\end{aligned}$$

Therefore, $\hat{V}_\pi$ is a fixed point of the Bellman operator $\mathcal{T}^\pi$, and thus equal to V_π.

Next, as we know that $Q_\pi(\cdot, a) = R_a + \gamma\langle P_a, V_\pi\rangle$, and $\Phi\hat{P}_a = P_a\Phi, \Phi\hat{R}_a = R_a$, we can obtain

$$\begin{aligned}
Q_\pi(\cdot, a) &= R_a + \gamma\langle P_a, V_\pi\rangle \\
&= \Phi\hat{R}_a + P_a\Phi\omega \\
&= \Phi\hat{R}_a + \Phi\hat{P}_a\omega \\
&= \Phi(\hat{R}_a + \hat{P}_a\omega)
\end{aligned}$$

Therefore, for any $\pi \in \Pi_\phi^D$, Q_π can be represented by a linear approximation operator on Φ, and thus Φ is linearly sufficient for learning.

□

B.4 PROOF OF THEOREM 7

Proof of Theorem 7. Lemma 2 in Gelada et al. (2019) is based on one policy in the induced MDP and bounded model errors. We can replace the Wasserstein distance $\mathcal{W}(\phi P(\cdot|o,a), \tilde{P}(\cdot|\phi(o),a))$ by the Euclidean distance $\|\phi P(o,a), \tilde{P}(\phi(o),a)\|$ as we focus on deterministic transitions.

For any policy $\pi \in \Pi_\phi^D$ that can be written as $\tilde{\pi} \circ \phi$, we have

$$|Q_\pi(o,a) - \tilde{Q}_{\tilde{\pi}}(\phi(o),a)| \leq \frac{\epsilon_R + \gamma K_{\phi,V}\epsilon_P}{1-\gamma} \tag{14}$$

Therefore, ϕ is $(1-\gamma)^{-1}(\epsilon_R + \gamma K_{\phi,V}\epsilon_P)$-sufficient for learning M.

Combined with Lemma 4, we can obtain the bound in Theorem 7.

□

B.5 PROOF OF THEOREM 8

Proof of Theorem 8. First of all, if there exists $\phi^{(T)}$ satisfying $\hat{P}_a(\phi(o_i)) = P_a^{(T)}[i]\Phi^{(T)}$, $\hat{R}_a(\phi(o_i)) = R_a^{(T)}[i]$, $\forall o_i \in \mathcal{O}^{(T)}$, then it is sufficient for the dynamics of the target task, and thus sufficient for learning the target task as stated in Proposition 6. Therefore, our focus is to show the existence of such a representation.

As $\phi^{(S)}$ is sufficient for $P_a^{(S)}$ and $R_a^{(S)}$ for all $a \in \mathcal{A}$, we have

$$\hat{P}_a(\phi^{(S)}(o^{(S)})) = P^{(S)}(o^{(S)},a)\Phi^{(S)} = \phi^{(S)}(P^{(S)}(o^{(S)},a)) \tag{15}$$

$$\hat{R}_a(\phi^{(S)}(o^{(S)})) = R^{(S)}(o^{(S)},a) \tag{16}$$

where we let $P^{(S)}(o^{(S)},a)$ denote the next state of $(o^{(S)},a)$, given that P is deterministic.

Based on Assumption 1, we know that there exists a function f such that $\forall o^{(T)} \in \mathcal{O}^{(T)}$, $f(o^{(T)}) \in \mathcal{O}^{(S)}$, and $f(P^{(T)}(o^{(T)},a)) = P^{(S)}(f(o^{(T)}),a)$, $R^{(T)}(o^{(T)},a) = R^{(S)}(f(o^{(T)}),a)$. Hence, we can obtain

$$\hat{P}_a(\phi^{(S)}(f(o^{(T)}))) = \phi^{(S)}(P^{(S)}(f(o^{(T)}),a)) = \phi^{(S)}(f(P^{(T)}(o^{(T)},a))) \tag{17}$$

$$\hat{R}_a(\phi^{(S)}(f(o^{(T)}))) = R^{(S)}(f(o^{(T)}),a) = R^{(T)}(o^{(T)},a) \tag{18}$$

Let $\hat{\phi}^{(T)} := \phi^{(S)} \circ f$, then we get

$$\hat{P}_a(\hat{\phi}^{(T)}(o^{(T)})) = \hat{\phi}^{(T)}(P^{(T)}(o^{(T)},a)) \tag{19}$$

$$\hat{R}_a(\hat{\phi}^{(T)}(o^{(T)})) = R^{(T)}(o^{(T)},a) \tag{20}$$

Therefore, $\hat{\phi}^{(T)}$ is a feasible solution satisfying model sufficiency in the target task, and thus is sufficient for learning.

Theorem 8 holds since we have shown (1) all feasible solutions to $\Phi^{(T)}\hat{P}_a = P_a^{(T)}\Phi$ and $\Phi^{(T)}\hat{R}_a = R_a^{(T)}$ are sufficient for learning in $M^{(T)}$, and (2) there exists at least one feasible solution to $\Phi^{(T)}\hat{P}_a = P_a^{(T)}\Phi$ and $\Phi^{(T)}\hat{R}_a = R_a^{(T)}$.

□

C REPRESENTATION LEARNING FOR APPROXIMATED VALUE ITERATION

Now we illustrate how our transfer algorithm and the proposed model-based regularization work for approximated value iteration. We focus on the case where the reward function $R(o,a) \geq 0$ for all $o \in \mathcal{O}$ and $a \in \mathcal{A}$.

Preliminaries The bases of value iteration is the Bellman optimality operator $\mathcal{T}^*$. For the value function, we have

$$\mathcal{T}^*V(o) = \max_{a\in\mathcal{A}}[R(o,a) + \gamma \sum_{o\in\mathcal{O}} P(o'|o,a)V(o')] \tag{21}$$

For the Q function, we have

$$\mathcal{T}^*Q(o,a) = R(o,a) + \gamma \sum_{o\in\mathcal{O}} P(o'|o,a) \max_{a'\in\mathcal{A}} Q(o',a'), \tag{22}$$

where we slightly abuse notation and use $\mathcal{T}^*$ for both the value function and the Q function when there is no ambiguity.

Starting from some initial Q_0 (or V_0) and iteratively applying $\mathcal{T}^*$, i.e., $Q_{k+1} = \mathcal{T}^* Q_k$, Q_k (or V_k) can finally converge to Q^* or V^* when $k \to \infty$, which is known as value iteration. When an approximation operator $\mathcal{H}$ is used, the process is called approximate value iteration (AVI): $Q_{k+1} = \mathcal{H}_\phi \mathcal{T}^* Q_k$. A prior work has shown the following asymptotic result:

Lemma 12 (Approximate Value Iteration for Q)**.** *For a reinforcement learning algorithm based on value iteration with approximation operator $\mathcal{H}$, if $\|\mathcal{H}\mathcal{T}^* Q_k - \mathcal{T}^* Q_k\|_\infty \leq \epsilon$, for all Q_k along the value iteration path, then we have*

$$\limsup_{k\to\infty} \|V^* - V_{\pi_k}\| \leq \frac{2\epsilon}{(1-\gamma)^2}. \tag{23}$$

Value Iteration Learning with Given Representation Given a representation mapping ϕ, we aim to learn an approximation function $h : \phi(\mathcal{O}) \to \mathbb{R}^{|A|}$ such that $\hat{Q}(o, \cdot) = h(\phi(o)) \approx Q(o, \cdot)$. For notation simplicity, we further use $h(\phi(o), a)$ to denote the approximated Q value $\hat{Q}(o, a)$ for $a \in \mathcal{A}$. We start from an initial h_0 that has a uniform value c for all inputs, where $c > 0$ can be randomly selected. The initial approximation for Q value is $\hat{Q}_0 = h_0 \circ \phi$. Then, at iteration $k > 0$, we solve $\hat{Q}_k = \mathcal{H}_\phi \mathcal{T}^* \hat{Q}_{k-1} = h_k \circ \phi$, where $h_k := \text{argmin}_h \|h \circ \phi - \mathcal{T}^* \hat{Q}_{k-1}\|_\infty$. We use a neural network (universal function approximator) to parameterize h, so the approximation error $\|h \circ \phi - \mathcal{T}^* \hat{Q}_{k-1}\|_\infty$ depends on the representation quality of ϕ.

Therefore, a representation mapping ϕ is ϵ-sufficient for learning with value iteration if $\|\mathcal{H}_\phi \mathcal{T}^* Q_k - \mathcal{T}^* Q_k\|_\infty \leq \epsilon$. Next, we identify the relationship between policy-independent model sufficiency and the learning sufficiency with value iteration methods.

Guaranteed Learning with Model-regularized Representation We first make the following assumption for the learned approximation function h.

Assumption 13 (Lipschitz Value Approximation)**.** *There exists a constant $K_{\phi,h}$, such that $\forall k \geq 0, o_1, o_2 \in \mathcal{O}, a \in \mathcal{A}$,*

$$|h_k(\phi(o_1), a) - h_k(\phi(o_2), a)| \leq K_{\phi,h} \|\phi(o_1) - \phi(o_2)\|, \tag{24}$$

where h_k is the approximation function in the k-th iteration.

Then, the following Theorem holds, which justifies that learning with model-regularized representation helps with value iteration learning.

Theorem 14. *For an MDP M, if encoder ϕ satisfies $\max_{o\in\mathcal{O}, a\in\mathcal{A}} |R(o, a) - \hat{R}_a(\phi(o))| \leq \epsilon_R$ and $\max_{o\in\mathcal{O}, a\in\mathcal{A}} \|\mathbb{E}_{o'\sim P(\cdot|o,a)} \phi(o') - \hat{P}_a(\phi(o))\|_2 \leq \epsilon_P$ for dynamics models $(\hat{P}_a, \hat{R}_a)_{a\in\mathcal{A}}$, then the approximated value iteration with approximation operator $\mathcal{H}_\phi$ under Assumption 13 satisfies*

$$\limsup_{k\to\infty} \|V^* - V^{\pi_k}\|_\infty \leq \frac{2}{(1-\gamma)^2} (\epsilon_R + \gamma \epsilon_P K_{\phi,h}). \tag{25}$$

Proof of Theorem 14. Let h_k be the approximation function in the k-th iteration. That is, the approximated Q function in the k-th iteration is $\hat{Q}_k = h_k \circ \phi$. As the rewards of all state-action pairs are non-negative, we have $h_k(\phi(o), a) \geq 0$.

Define a function $\hat{h}_{k+1}$ as

$$\hat{h}_{k+1}(\phi(o), a) = \hat{R}_a(\phi(o)) + \gamma \max_{a'\in\mathcal{A}} h_k(\hat{P}_a(\phi(o)), a') \tag{26}$$

Given that

$$\mathcal{T}^* \hat{Q}_k(o, a) = R(o, a) + \gamma \mathbb{E}_{o'\sim P(\cdot|o,a)} [\max_{a'\in\mathcal{A}} \hat{Q}_k(o', a')] \tag{27}$$

$$= R(o, a) + \gamma \mathbb{E}_{o'\sim P(\cdot|o,a)} [\max_{a'\in\mathcal{A}} h_k(\phi(o'), a')], \tag{28}$$

we have that for any $o \in \mathcal{O}$, $a \in \mathcal{A}$,

$$\left|\hat{h}_{k+1}(\phi(o), a) - \mathcal{T}^* \hat{Q}_k(o, a)\right| \tag{29}$$

$$= \left|\hat{R}_a(\phi(o)) + \gamma \max_{a' \in \mathcal{A}} h_k(\hat{P}_a(\phi(o)), a') - (R(o, a) + \gamma \mathbb{E}_{o' \sim P(\cdot|o,a)}[\max_{a' \in \mathcal{A}} h_k(\phi(o'), a')])\right| \tag{30}$$

$$\leq \left|\hat{R}_a(\phi(o)) - R(o, a)\right| + \gamma \left|\max_{a' \in \mathcal{A}} h_k(\hat{P}_a(\phi(o)), a') - \mathbb{E}_{o' \sim P(\cdot|o,a)}[\max_{a' \in \mathcal{A}} h_k(\phi(o'), a')]\right| \tag{31}$$

$$\leq \epsilon_R + \gamma \mathbb{E}_{o' \sim P(\cdot|o,a)}[\left|\max_{a' \in \mathcal{A}} h_k(\hat{P}_a(\phi(o)), a') - \max_{a' \in \mathcal{A}} h_k(\phi(o'), a')\right|] \tag{32}$$

$$\leq \epsilon_R + \gamma \mathbb{E}_{o' \sim P(\cdot|o,a)} \max_{a' \in \mathcal{A}} \left|h_k(\hat{P}_a(\phi(o)), a') - h_k(\phi(o'), a')\right| \tag{33}$$

$$\leq \epsilon_R + \gamma \mathbb{E}_{o' \sim P(\cdot|o,a)} \max_{a' \in \mathcal{A}} K_{\phi,h} \left\|\hat{P}_a(\phi(o) - \phi(o')\right\| \tag{34}$$

$$\leq \epsilon_R + \gamma \epsilon_P K_{\phi,h}, \tag{35}$$

where (33) is due to the non-negativity of h_k, and (34) is due to Assumption 13.

Now we have shown that the constructed $\hat{h}_{k+1}$ satisfies

$$\|\hat{h}_{k+1} \circ \phi - \mathcal{T}^* \hat{Q}_k\|_\infty \leq \epsilon_R + \gamma \epsilon_P K_{\phi,h}. \tag{36}$$

According to the definition of $\mathcal{H}_\phi$, we obtain

$$\|h_{k+1} \circ \phi - \mathcal{T}^* \hat{Q}_k\|_\infty \leq \|\hat{h}_{k+1} \circ \phi - \mathcal{T}^* \hat{Q}_k\|_\infty \leq \epsilon_R + \gamma \epsilon_P K_{\phi,h} \tag{37}$$

since $\mathcal{H}_\phi$ finds a h_{k+1} that minimizes the approximation error.

Therefore, Theorem 14 follows by combining Inequality (37) and Lemma 12.

□

D Experiment Details and Additional Results

D.1 Implementation Details

Environment Settings in MuJoCo Target Tasks For the target task of MuJoCo environments, we add the center of the mass based inertia and velocity into the observations of the agent, concatenating them with the original observation on the source task. Consequently, in the target environments, the dimensionality of the observation space on target task become much larger than that of the source task. On Hopper, the dimensionality of the target observation is 91, whereas the the source observation space only has 11 dimensions. The dimensionalities of target tasks on HalfCheetah, Hopper and Walker are 145, 145, 91 respectively.

Implementation DQN For the CartPole target task, we extract the screen of the environment, and crop the pixel input to let the image be centered at the cart. [1] The DQN network has a 2-layer encoder and a 2-layer Q head, with the representation dimension set as 16. We use the Adam optimizer with learning rate 0.001. The target Q network is updated every 10 iterations. We let the dynamics models $\hat{P}_a$ and $\hat{R}_a$ be linear, so that the representation be more informative in terms of representing the transition dynamics.

Transfer SAC We conduct the experiments of transfer SAC on MuJoCo-v3 using TITAN RTX GPU. The number of hidden units for all neural networks is 256. The state encoder and the two action encoders have a two-layer architecture and the output feature size is 256. The activation function is ReLU and the learning rate is $3 \cdot 10^{-4}$. The regularization weight is set as 1. We train the dynamics model and the reward model every 50k interactive steps in the source task.

[1] The pre-processing of inputs follows https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html.

D.2 ADDITIONAL ABLATION STUDY

Figure 5 shows the ablation study of our method in continuous tasks. We compare our method with the following variants:
(1) learning auxiliary tasks without transfer,
(2) only transferring transition models $\hat{P}$ and
(3) only transferring reward models $\hat{R}$.

Compared with the single-task learning baseline (the blue curves), we find that all the variants of our method can make some improvements, which suggests that learning dynamics models as auxiliary tasks, transferring $\hat{P}$ and $\hat{R}$ are all effective designs for accelerate the target task learning. Finally, our method (the red curves) that combines the above components achieves the best performance, justifying the effectiveness of our transfer algorithm.

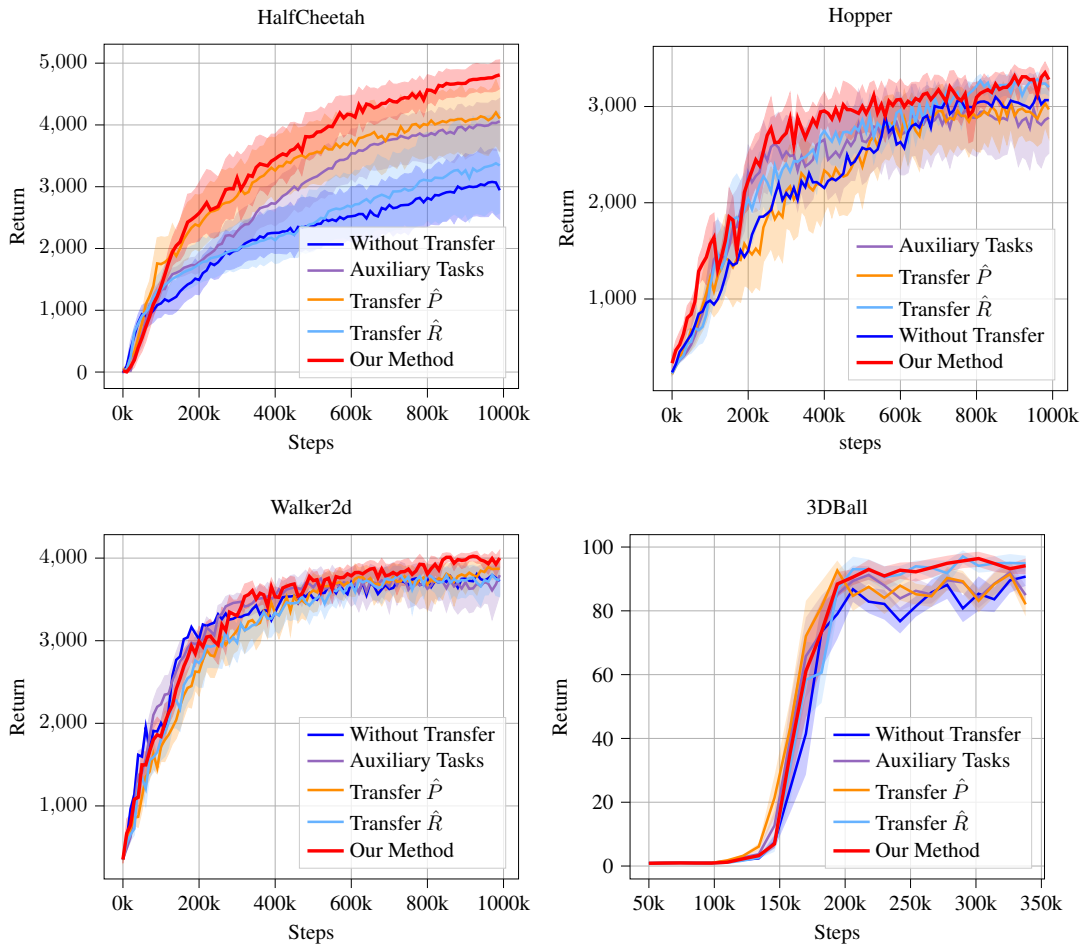


Figure 5: Ablation study of our method with SAC on target tasks.