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 usually fails due to the mismatch of input features, and therefore one has to train another policy from scratch, which is computationally and sample inefficient. In this paper, we propose a novel algorithm that extracts the latent-space dynamics in the source task, and transfers the dynamics model to the target task with a model-based regularizer. Theoretical analysis shows that the transferred dynamics model helps with representation learning in the target task. 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 have justified that our algorithm significantly improves the efficiency and stability of learning in the target task.

1 INTRODUCTION

Deep Reinforcement Learning (DRL) is widely used in many large-scale applications which are under rapid evolution and development, such as robotics, gaming and auto-driving. Therefore, it is crucial to improve the efficiency of training DRL agents by 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 and different dynamics/reward, such that the source policy can adapt to the target task by learning task-specific skills while keeping task-agnostic skills. However, these approaches do not apply to the case where the observation space changes (e.g. the dimensionality changes) while the dynamics remain similar.

Observation change is common in practice due to hardware upgrading, data restriction, curriculum design, etc. 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 (e.g., 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 time- and sample-consuming, 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 a challenging setting where the observation change is
(1) unpredictable: there is no prior knowledge about how the observation changes. Many works (Taylor et al., 2007; Mann & Choe, 2013) require an inter-task mapping as the prior knowledge, but such a mapping might not be available in practice.
(2) drastic: the source and target tasks have significantly different observation feature spaces. Raiman et al. (2019) propose to selectively adjust the network weights when there are additional input features, which does not work when the change is drastic (e.g. vector to image).

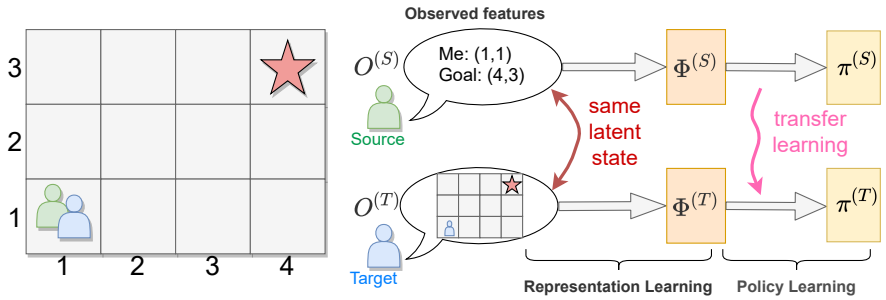


Figure 1: An example of the transfer problem with changed observation space. The source-task agent observes the x-y coordinates of itself and the goal, while the target-task agent observes a top-down view/image of the whole maze. The two observation spaces are drastically different, but the two tasks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

(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.

To remedy the above challenges and achieve knowledge transfer, we make a key observation that the source task and the target task have the same dynamics in a latent state space (e.g. in Figure 1, $O^{(S)}$ and $O^{(T)}$ 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 using 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.

Summary of Contributions. (1) To the best of our knowledge, we are the first to identify 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 is a good representation" and analyze the sufficient conditions for a representation being good. (3) Based on theoretical insights, we propose a novel algorithm that automatically transfers knowledge across observation representations with learned latent dynamics models. (4) Empirical study in various environments shows that our proposed algorithm significantly improves the learning performance of basic RL agents in the target task through effective knowledge transfer.

2 Preliminaries and Background

Basic Notations and Background for RL An RL task can be modeled by a Markov Decision Process (MDP), 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 from the environment, 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 value function V or the action function 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 $\pi_{k+1}(o) := \text{argmax}_{a \in \mathcal{A}} Q_{\pi_k}(o, a), \forall o \in \mathcal{O}$. 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^*.

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 or the value function over the representation space $\phi(\mathcal{O})$.

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 policy and the target policy have different input shapes.

In this work, as motivated in the introduction, we focus on the setting that the dynamics $(P^{(S)}, R^{(S)})$ and $(P^{(T)}, R^{(T)})$ of the two MDPs we transfer knowledge across are defined on different state/observation spaces but share some structural similarities. Specifically, the target task has a different observation space but there exists a mapping such that the transition dynamics under the mapping in the target task share the same transition dynamics in the source task, as will be made rigorous in Assumption 1 below.

Assumption 1. *There exist 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 simple down-sampling function. (2) Although Assumption 1 assumes the existence of such an underlying mapping between the source and the target observations, our algorithm does not assume any prior knowledge of the mapping f. (3) f is a general function without extra restrictions. f can be a many-to-one mapping, i.e., more than one observations in the target task 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 to the learner. To address this problem, in the next section we propose to learn a latent representation of the observations and transfer dynamics on the latent representation to speed up the learning in the target task, without learning 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. We first formally characterize the properties of representation learning for RL in Section 4.1, which is crucial to motivate our algorithm, then introduce our proposed algorithm for across-observation transfer in Section 4.2. Theoretical analysis for the proposed algorithm is provided in Section 4.3.

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 finding an optimal policy. However, it is not well understood yet what properties a good representation should have for an RL task. Before introducing our algorithm that encourages the agent to

learn a better representation with knowledge transfer, we first formulate the conditions for a good representation and the relation between representation and learning performance.

Given a representation mapping ϕ, the approximation of the Q value for any $o \in \mathcal{O}, a \in \mathcal{A}$ can be 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. Define an *approximation operator* $\mathcal{F}_\phi$ of the Q value such that $\mathcal{F}_\phi Q(o) := f(\phi(o);\Theta^*)$, where $\Theta^* = \text{argmin}_\Theta \mathbb{E}[\|f(\phi(o);\Theta) - Q(\phi,\cdot)\|]$. The approximation error $\|Q - \hat{Q}\|$ depends on both the selection of the representation ϕ and the selection of approximation operator.

The quality of the encoder ϕ is crucial for learning an accurate value function or learning a good policy. An ideal encoder ϕ should discard the irrelevant information in the raw observation while keeping the most essential messages, then make it easy for learning a policy in the resulted representation space. 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 Y corresponding to each input X. 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 related to the policy values, and is *policy-dependent*. Below we formally characterize the sufficiency of a representation mapping in terms of a fixed policy and in terms of 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 the MDP is reduced into a Markov Reward Process (MRP), where the optimality of representation is studied in some literature (Parr et al., 2008; Lyle et al., 2021). In Definition 9, Appendix A, we propose a formal definition that a representation is sufficient for a fixed policy π as long as it can encode the policy value V_π.

However, the goal of RL is to search for an optimal policy in the environment. Therefore, it is important to investigate the conditions for a representation to be sufficient for learning. Intuitively, a representation mapping is sufficient for learning if we are able to find an optimal policy over the representation space $\phi(\mathcal{O})$. Therefore, the representation should be sufficient for multiple policies that are important for learning. 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, $\Pi_\phi^D := \{\pi \in \Pi : \exists \tilde{\pi} : \phi(\mathcal{O}) \to \mathcal{A}$ 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 policies that are parameterized in the representation space. Now we proceed to 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 $\mathcal{F}_\phi$ if $\mathcal{F}_\phi Q_\pi = Q_\pi$, for all $\pi \in \Pi_\phi^D$. (ϕ is **ϵ-sufficient** for learning if $\|\mathcal{F}_\phi Q_\pi - Q_\pi\| \leq \epsilon$, **linearly-sufficient** if $\mathcal{F}_\phi$ is linear projection.)*

Definition 3 suggests that the representation is sufficient for learning a near-optimal policy as long as it is sufficient for policies in Π_ϕ^D. Then, the lemma below justifies that a representation satisfying Definition 3 can ensure that reinforcement learning algorithms converge 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 $\mathcal{F}_\phi$ 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 based on 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. How can we enforce the networks to learn a good representation? Following Definition 3, a natural method is to let the representation fit as many policy values as possible as auxiliary tasks, which matches the ideas in many related works. For example, Bellemare et al. (2019) propose to fit a set of representative policies (called *adversarial value functions*). But the exact solution is computationally intractable since there are an exponential number of policies to consider. Dabney et al. (2020) choose to fit the values of all past policies (along the *value improvement path*), which requires less computation resource. But during learning, the future policies in the value improvement path are unknown, thus the requirements for a representation being good are not fully satisfied.

Different from these works that directly fitting value functions of multiple policies, we propose a computationally efficient way to realize sufficient representation for learning: *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.

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 adapts to the target task more easily than normal learning methods. 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 representation (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.

For both the source and the target task, we use transition loss L_P and reward loss L_R to regularize the learned representation. 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, where the policy/value and the dynamics model share the same encoder. Whether to learn policy network or value network depends on the base DRL algorithm. For example, if DQN is used, then a Q-value head should be appended to the representation z.

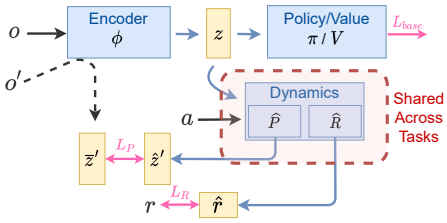


Figure 2: The architecture of proposed method. $\hat{P}$ and $\hat{R}$ are learned in the source task, then transferred to the target task and fixed during training.

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 $\nabla_{\hat{P}} L_P(\hat{P}; \hat{\phi}^{(S)})$ and $\nabla_{\hat{R}} L_R(\hat{R}; \hat{\phi}^{(S)})$, 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 this section, we analyze the theoretical insights behind our algorithm design. By associating model sufficiency and learning sufficiency, we show theoretical justifications and practical advantages of our proposed method in terms of transfer learning.

For notation simplicity, in this section we assume that the original observation space $\mathcal{O}$ is finite with cardinality n, and the transition dynamics P is deterministic, i.e., $supp(P(o, a)) = 1$ where $supp(\cdot)$ is the support of a vector. We let Φ denote the representation matrix, where the i-th row of Φ refers to the feature of the i-the observation. We use $P_a \in \mathbb{R}^{n \times n}$, $R_a \in \mathbb{R}^n$ to denote the transition matrix and the reward vector of all states under action a, respectively. That is, $P_a[i, j] = P(o_j | o_i, a)$, $R_a[i] = R(o_i, a)$, where $[i, j]$ refers to the entry in the i-th row and the j-th column. Note that the dynamics (P_a, R_a) are policy-independent. The following definition describes the quality of a representation map through the lens of dynamics.

Definition 5 (Policy-independent Model Sufficiency)**.** *A representation mapping ϕ is sufficient for a dynamics pair (P_a, R_a) if $\forall a \in \mathcal{A}$ there exists operators $\hat{P}_a : \mathbb{R}^d \to \mathbb{R}^d$ and $\hat{R}_a : \mathbb{R}^d \to \mathbb{R}$ such that $\forall o_i \in \mathcal{O}$, $\hat{P}_a \phi(o_i) = P_a[i]\Phi$, $\hat{R}_a \phi(o_i) = R_a[i]$, where $P_a[i] := P_a[i, :]$.*

Model Sufficiency Induces Learning Sufficiency What is the relationship between the model sufficiency and the sufficiency for learning (Definition 3)? Proposition 6 below shows that learning sufficiency can be achieved via ensuring model sufficiency, which justifies that learning the latent dynamics models as an auxiliary task helps with the representation quality.

Proposition 6 (Learning Sufficiency Induced by Policy-independent Model Sufficiency)**.** *For an MDP M with dynamics P, R, if ϕ is sufficient for P_a, R_a, $\forall a \in \mathcal{A}$, then it is sufficient (but not linearly) for learning in M.*

Proposition 6 proven in Appendix B.2 suggests learning prediction models of the next state representation and the immediate reward as auxiliary tasks, as we propose in Algorithm 1 and 2. It is important to note that the representation regularized by the policy-independent models may not be linearly sufficient for learning. Therefore, we set the policy head after the latent representation to have more than one layers to reduce approximation error.

If model fitting is not perfect but with bounded error, the learned representation is still nearly sufficient for learning. Theorem 7 below shows a general guarantee for learning with model-regularized representation. The proof of Theorem 7 is in Appendix B.4, where we combine Lemma 4 and a variant of the value difference bound derived by Gelada et al. (2019).

Theorem 7. *For an MDP M, if encoder ϕ satisfies $\max_{o\in\mathcal{O},a\in\mathcal{A}} |R(o,a) - \phi(o)\hat{R}_a| \leq \epsilon_R$ and $\max_{o\in\mathcal{O},a\in\mathcal{A}} \|\mathbb{E}_{o'\sim P(\cdot|o,a)}\phi(o') - \hat{P}_a(\phi(o))\|_2 \leq \epsilon_P$, then the approximated policy iteration with approximation operator $\mathcal{F}_\phi$ starting from any initial policy $\pi_0 \in \Pi^D_\phi$ satisfies,*

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

where κ_V is an upper bound of the value Lipschitz constant as defined in Appendix B.4.

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 $M^{(T)}$. Suppose $\Phi^{(S)}$ is sufficient for $P^{(S)}_a, R^{(S)}_a$ with operators $\hat{P}_a, \hat{R}_a$, then a representation $\Phi^{(T)}$ is sufficient for learning in $M^{(T)}$ if it satisfies $\Phi^{(T)}\hat{P}_a = P^{(T)}_a\Phi$ and $\Phi^{(T)}\hat{R}_a = R^{(T)}_a$.*

Why Non-linear Policy Head As discussed above, fitting policy-independent dynamics encourages the representation to be sufficient for learning, though not linearly sufficient. Is it possible to achieve linear sufficiency via fitting latent dynamics? In Appendix A, we show that 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 dynamics models induced by policy π. However, this method is computationally expensive and inefficient for knowledge transfer, as we have to learn and store a large number of latent dynamics models. Therefore, there is a trade-off between approximation complexity and representation complexity. Learning a linearly sufficient representation is good, as it 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.

5 Related Work

Transfer RL across Observation Representations 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.

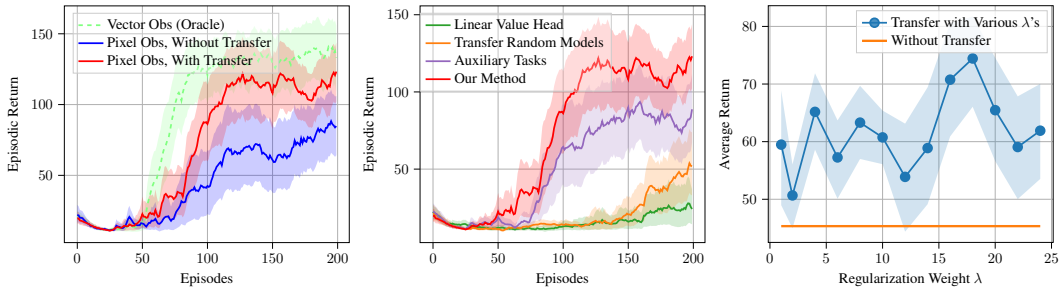
Representation Learning in RL In environments with rich observations, representation learning is crucial for the efficiency of RL methods. Learning unsupervised auxiliary tasks (Jaderberg et al., 2016) is shown to be effective for learning a good representation. The relationship between learning policy-dependent 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 empirically 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) propose to 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 methods. However, the key messages for solving the RL task may be ignored since environment dynamics are not considered during pre-training. Our algorithm can be combined with contrastive learning techniques to further improve learning performance in the target task.

6 Experiment

In this section, we empirically evaluate our proposed transfer learning algorithm in a wide range of environments and various scenarios of observation changes. The detailed setup and hyperparameters are recorded in Appendix C.

Transfer Knowledge from Vector Observations to Pixel Observations To verify that our transfer algorithm works for drastic observation changes, we first focus on one of the most challenging scenarios 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 with the cart position, cart velocity, pole angle and pole angler velocity, while in the target task, the agent observes a screenshot of the rendered environment. Figure 3a shows a comparison of performance between an agent with our transfer algorithm and 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 Proposition 6. 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



(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) Additional 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 $\lambda = 0$).

our proposed transfer algorithm. These empirical results have verified our theoretical insights and shown 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 C for details). 3DBall is a game provided by Unity (Juliani et al., 2018), which has two different observation specifications that naturally fit our transfer setting: The source observation has 8 features corresponding to the rotation of the agent cube, and the position and velocity of the ball. The target observation (harder version) has a stack of 9 past frames, each of which corresponds to the rotation of the agent cube, and the position of the ball. 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 C, 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 transferring $\hat{P}$ 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 (with low errors). More techniques of improving model-based learning, such as bisimulation (Zhang et al., 2020), can be applied to further improve the transfer performance.

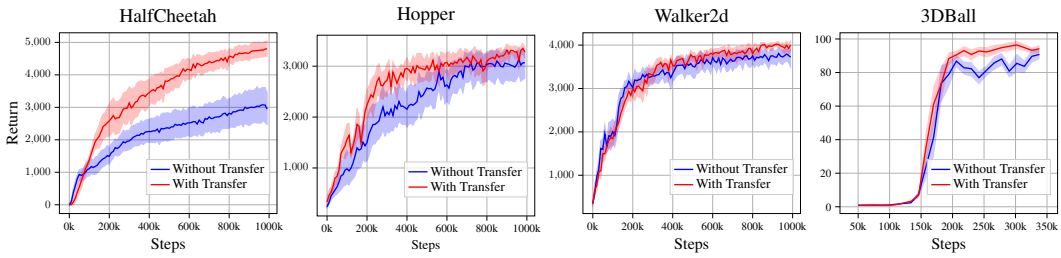


Figure 4: Comparison of single-task SAC and SAC combined with our transfer method in the target task.

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.

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.

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.

Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 752–759, 2008.

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.