Invariant Action Effect Model for Reinforcement Learning

Zheng-Mao Zhu[1,2*], **Shengyi Jiang**[1], **Yu-Ren Liu**[1], **Yang Yu**[1,2], **Kun Zhang** [3]

[1] National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu, China
[2] Peng Cheng Laboratory, Shenzhen, Guangdong, China
[3]Department of Philosophy, Carnegie Mellon University,Pittsburgh, PA, United States
{zhuzm, jiangsy, liuyr, yuy}@lamda.nju.edu.cn, kunz1@cmu.edu

Abstract

Good representations can help RL agents perform concise modeling of their surroundings, and thus support effective decision-making in complex environments. Previous methods learn good representations by imposing extra constraints on dynamics. However, in the causal perspective, the causation between the action and its effect is not fully considered in those methods, which leads to the ignorance of the underlying relations among the action effects on the transitions. Based on the intuition that the same action always causes similar effects among different states, we induce such causation by taking the invariance of action effects among states as the relation. By explicitly utilizing such invariance, in this paper, we show that a better representation can be learned and potentially improves the sample efficiency and the generalization ability of the learned policy. We propose **I**nvariant **A**ction **E**ffect **M**odel (IAEM) to capture the invariance in action effects, where the effect of an action is represented as the residual of representations from neighboring states. IAEM is composed of two parts: (1) a new contrastive-based loss to capture the underlying invariance of action effects; (2) an individual action effect and provides a self-adapted weighting strategy to tackle the corner cases where the invariance does not hold. The extensive experiments on two benchmarks, i.e. Grid-World and Atari, show that the representations learned by IAEM preserve the invariance of action effects. Moreover, with the invariant action effect, IAEM can accelerate the learning process by 1.6x, rapidly generalize to new environments by fine-tuning on a few components, and outperform other dynamics-based representation methods by 1.4x in limited steps.

Introduction

Despite recent progress on deep reinforcement learning, sample-efficiency and generalization ability are still big challenges in achieving a satisfactory performance, especially in environments with high-dimensional observation space (Mnih et al. 2013, 2015). It has been proved both theoretically and empirically that learning policies from low-dimensional features can be much more sample-efficient than directly learning. A good representation can also improve generalizability (Silver et al. 2016; Tassa et al. 2018). Such advantages motivate learning an abstract representation by encoding high-dimensional states into compressed representations while retaining the most important information for learning tasks (Biza et al. 2020).

Recent works (Srinivas, Laskin, and Abbeel 2020; Zhang et al. 2020b) use self-supervised methods to learn abstract states of visual observations, which can be divided into two major categories: reconstruction-based and dynamics-based approaches. Reconstruction-based approaches learn representations by restoring the raw observations, but such representations can be easily distracted by large amounts of irrelevant information in observations(Higgins et al. 2017). Dynamics-based approaches (Biza et al. 2020) filter the irrelevant information by grouping states that are indistinguishable w.r.t state sequences given any action sequences tested. However, in the causal perspective, the causation between the action and its effect is ignored since those approaches learn action effects on each state individually. This causation implies the underlying relations of action effects in the representation space, which benefits the sample efficiency and generalizability.

Based on the intuition that the same action always causes similar effects among different states, we propose IAEM (Invariant Action Effect Model) to utilize this causation in the form of an invariance relation between action effects into learning latent dynamics, where the effect of an action is represented as the residual between the representations of neighboring states. IAEM learns the invariance property for RL environments with two components: (1) A new contrastive loss by viewing the action effect from the same action as the positive samples and others as negative samples. We show that with this contrastive loss, the invariance of the action effect can be preserved in the residuals of the state representations. (2) An individual action effect and a learnable weight for using the invariant action effect in practice, which can reduce the discrepancy between the ideal invariance property and the non-invariance in some marginal conditions (e.g., when an agent moves towards a barrier, it will stay put instead of moving through the barrier).

In summary, this paper makes the following key contributions:

- The IAEM is the first to learns an **action effect disentangled** representation, which forming a better representation for learning policies and interpretability.

*This work is supported by National Key Research and Development Program of China (2020AAA0107200), and NSFC(61876077).

- The IAEM outperforms baseline dynamics-based methods by showing **20% performance gains** in Atari games in limited 30M environment steps.
- The IAEM saves up to **40% samples** compared with baseline dynamics-based methods while reaching the converged performance.

Related Work

Learning representation in RL has been widely studied in high-dimensional observation settings. Reconstruction loss, such as enforcing abstract states to recover observations (Igl et al. 2018), is one of the most common constraints used to guide the learning process (Corneil, Gerstner, and Brea 2018; Hafner et al. 2019; Feng et al. 2021b). The main disadvantage of reconstruction constraint is training the decoder, which is time-consuming and usually not necessary for decision-making tasks (van der Pol et al. 2020). Dynamics-based representation learning, on the other hand, aims to preserve the dynamics-related property in the latent space. Self-supervised representation learning aims to group states together without loss of that property (Anand et al. 2019; Kipf, van der Pol, and Welling 2020). To avoid erroneously grouping states, Gelada et al. (2019) includes both the transition function and the reward function in grouping states; François-Lavet et al. (2019) introduces an approximate entropy maximization penalty for abstract representations. van der Pol et al. (2020) defines action-equivalence for better recovering the action effect in the latent space. However, these prevailing dynamics-based approaches do not utilize the underlying relations of the action effects, which leads to repeated learning of similar action effects. To solve this problem, our IAEM represents the action effect as the residual representations of neighboring states and uses contrastive learning to learn invariant action effects.

Contrastive-based representation learning in RL. Contrastive learning is a framework to learn representations organized by similarity and dissimilarity. In reinforcement learning, structuring positive and negative samples for contrastive learning is a challenge. CURL (Srinivas, Laskin, and Abbeel 2020) directly takes a state and its augmentations as positives, but it didn't utilize the dynamics information in the samples. TCN and its variants (Sermanet et al. 2018; Dwibedi et al. 2018) use an image and itself in another view as positive, which is similar to CURL to some extend. Previous works do not utilize the dynamics information in RL, which is an in-negligible feature of data in RL ((Feng et al. 2021a; Zhang et al. 2021)). In contrast, our IAEM considers the dynamics in defining positives and negatives, i.e. we define the residual between two states to be positives or negatives according to the action.

Preliminaries

Markov Decision Process (MDP). We describe the RL environment as an MDP with five-tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ (Bellman 1957), where $\mathcal{S}$ is a finite set of states; $\mathcal{A}$ is a finite set of actions; P is the transition function with $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$ denoting the next-state distribution after taking action $\mathbf{a}$ in state $\mathbf{s}$; R is a reward function with $R(\mathbf{s}, \mathbf{a})$ denoting the expected immediate reward gained by taking action a in state s; and $\gamma \in [0, 1]$ is a discount factor. An agent chooses actions $\mathbf{a}$ according to a policy $\mathbf{a} \sim \pi(\mathbf{s})$, which updates the system state $\mathbf{s}' \sim P(\mathbf{s}, \mathbf{a})$, yielding a reward $r \sim R(\mathbf{s}, \mathbf{a})$. The agent's goal is to maximize the the expected cumulative return by learning a good policy $\max_{\pi, P} \mathbb{E}[\gamma^t R(\mathbf{s}_t, \mathbf{a}_t)]$. The state-action value Q_π of a policy π is the expected discounted reward of executing action a from state $\mathbf{s}$ and subsequently following policy π: $Q_\pi(\mathbf{s}, \mathbf{a}) := R(\mathbf{s}, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{s}' \sim P, \mathbf{a}' \sim \pi} [Q_\pi(\mathbf{s}', \mathbf{a}')]$.

Dynamics-based Representation Learning. The goal of dynamics-based representation learning is to learn representations preserving the dynamics. In this paper, we approach dynamics-based representation learning from the perspective of state abstractions. State abstractions can reduce the size of a given problem's state space by grouping together similar states, which maps the state space S to a compact representation space Z while preserving the reward and dynamics (Dean and Givan 1997a). Bi-simulation and bi-simulation metrics are different forms of state abstractions that group states according to "behaviorally equivalent" and "behaviorally similar" (e.g., how similar the future states and rewards are given the same action sequences). A generalization of the mapping induced by bi-simulation is the notion of MDP homomorphism (Randran and Barto 2004). MDP homomorphism were introduced by (Randran and Barto 2001) as an extension of (Dean and Givan 1997b). We use the definition of MDP homomorphism Given by (Randran and Barto 2004) as follows:

Definition 1 (Deterministic MDP Homomorphism) *The deterministic MDP homomorphism from a deterministic MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$ to another deterministic MDP $\bar{\mathcal{M}} = \langle \mathcal{Z}, \mathcal{A}_\mathcal{Z}, P_\mathcal{Z}, R_\mathcal{Z}, \gamma \rangle$ is a tuple $h = \langle f_\mathcal{S}, f_\mathcal{A} \rangle$. In this tuple, $f_\mathcal{S} : \mathcal{S} \to \mathcal{Z}$ is the state embedding function and $f_\mathcal{A} : \mathcal{A} \to \mathcal{A}_\mathcal{Z}$ is the action embedding function. $f_\mathcal{S}$ and $f_\mathcal{A}$ hold the following identities for all $\boldsymbol{s}, \boldsymbol{s}' \in \mathcal{S}$, $\boldsymbol{a} \in \mathcal{A} : P(\boldsymbol{s}, \boldsymbol{a}) = \boldsymbol{s}' \implies P_\mathcal{Z}(f_\mathcal{S}(\boldsymbol{s}), f_\mathcal{A}(\boldsymbol{a})) = f_\mathcal{S}(\boldsymbol{s}'), R_\mathcal{Z}(f_\mathcal{S}(\boldsymbol{s}), f_\mathcal{A}(\boldsymbol{a})) = R(\boldsymbol{s}, \boldsymbol{a})$.*

Given samples $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$ from the environment, recent works (François-Lavet et al. 2019; van der Pol et al. 2020) treat the MDP homomorphism as the following loss on the transition and reward functions: (1) The transition function preserving loss to minimize the distance between the predicted abstract states and sampled abstract states. A forward model $P_\mathcal{Z}$ is built to predict the next state representations based on the current state representations and chosen actions as $\mathcal{L}_{\text{transition}} = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})}[P_\mathcal{Z}(f_\mathcal{S}(\mathbf{s}_t), \mathbf{a}_t) - f_\mathcal{S}(\mathbf{s}_{t+1})]^2$. (2) The reward function preserving loss to predict the true rewards based on the abstract states and actions as $\mathcal{L}_{\text{reward}} = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, r_{t+1})}[R_\mathcal{Z}(f_\mathcal{S}(\mathbf{s}_t), \mathbf{a}_t) - r_t]^2$. (3) The mode collapse preventing loss to prevent trivial solutions, the distance between abstract states should be maximized as $\mathcal{L}_{\text{norm}} = \mathbb{E}_{(\mathbf{s}_t, \mathbf{s}_k)} \max(0, \epsilon - \|f_\mathcal{S}(\mathbf{s}_t) - f_\mathcal{S}(\mathbf{s}_k)\|^2)$, where $\mathbf{s}_k$ is a random sample.

In this paper, we use θ, ϕ and ζ to parameterize the abstraction function $f^\theta_\mathcal{S}$, the latent transition function and the latent reward function $R^\zeta_\mathcal{Z}$ respectively. We let $P^\phi_\mathcal{Z}$ be the form of $P^\phi_\mathcal{Z}(\mathbf{z}, \mathbf{a}) = \mathbf{z} + A^\phi_\mathcal{Z}(\mathbf{z}, \mathbf{a})$, and let $A^\phi_\mathcal{Z} : \mathcal{Z} \times \mathcal{A} \to \mathcal{A}_\mathcal{Z}$

be a function mapping from the state representation and the action into the action effect.

Invariant Action Effect Model

The Markov property in MDP guarantees that the state and the action are the only causal parents of the action effect and reward, which is represented by a graphical model in previous work(Zhang et al. 2020a). However, such causal modeling is insufficient since it does not fully consider the causation between the action and the action effect. In RL environments, there are several relations among different actions, including similarity and orthogonality, which also exist among the action effects due to this causation. In this work, we learn representations by building the constraints based on the relations, which requires fewer training data and improves the generalization.

Our method contains two steps: First, we define the relations of action effects based on the intuition that the same action often has a similar effect among different states. We call the action effect *invariant* if the action effects on different states but the same action are similar or even the same and take the invariance as the prior knowledge of effects relations. Second, we leverage MDP homomorphism to learn representations and design a new contrastive loss to keep the invariance of action effects, where action effects are the residual of representations. Without loss of generality, we consider deterministic MDPs with a discrete action space.

The statements of IAEM are organized in three steps: (1) Keep the action effect invariant. (2) Adapt the invariant action effect on marginal conditions. (3) Learn policies based on the representations with the invariant action effect. These three steps will be discussed in detail in the following subsections.

Learning Invariant Action Effects

To learn representations with the invariant action effect, we design a new contrastive loss for the action effects, which views the action effects from the same action as **positive samples** and those from different actions as **negative samples**. Given a set of experience tuples $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})\}_{t=1}^{N}$ by rolling out several initial policies, we first pick a batch $\mathcal{B} = \{(\mathbf{s}_t, \mathbf{a}_t), (\mathbf{s}_k, \mathbf{a}_k), (\mathbf{s}_m, \mathbf{a}_m)\} \sim \mathcal{D}$ where $\{(\mathbf{s}_t, \mathbf{a}_t), (\mathbf{s}_k, \mathbf{a}_k)\}$ are positive samples due to $\mathbf{a}_t = \mathbf{a}_k$ and $\{(\mathbf{s}_t, \mathbf{a}_t), (\mathbf{s}_m, \mathbf{a}_m)\}$ are negative samples due to $\mathbf{a}_t \neq \mathbf{a}_m$. Based on this batch, we denote the ***action effect similarity*** $g_{t,k} \triangleq g(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_k, \mathbf{a}_k)$ as follows:

$$g(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_k, \mathbf{a}_k) = A^{\phi}_{\mathcal{Z}}(f^{\theta}_{\mathcal{S}}(\mathbf{s}_t), \mathbf{a}_t)^T A^{\phi}_{\mathcal{Z}}(f^{\theta}_{\mathcal{S}}(\mathbf{s}_k), \mathbf{a}_k)/\tau. \tag{1}$$

With this similarity, we design our action contrastive loss based on the InfoNCE loss (van den Oord, Li, and nyals 2018) as:

$$\mathcal{L}_{\text{invariant}}(\theta, \phi) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t), (\mathbf{s}_k, \mathbf{a}_k), (\mathbf{s}_m, \mathbf{a}_m)}[-\log \frac{e^{g_{t,k}}}{e^{g_{t,k}} + e^{g_{t,m}}}]. \tag{2}$$

With this new loss, the action effects $A^{\phi}_{\mathcal{Z}}$ of the same actions will be similar and those of different actions will be different.

Previous contrastive-based approaches calculate the similarity g by treating its augmented data (Srinivas, Laskin, and Abbeel 2020) or its neighboring states (Hénaff et al. 2019) as the positive samples of a given state. In both cases, transition information of data in RL is not preserved. The absence of the transition information results in representations losing the dynamics-based relations among each other, which violates the goal of preserving related information and thus harms the ability for downstream tasks. By contrast, our contrastive loss captures the difference between neighboring states by taking neighboring states as sample pairs, which preserves the transition information. What is more, this contrastive loss makes action effects of the same actions close, which also preserves the relations of action effects. Representations with such relations of action effects have better generalization ability and are sample-efficient in training.

Adapting Invariant Action Effects

After defining how to learn an invariant action effect, we study how to adapt such action effect in practical environments. In many practical environments, there are some marginal conditions where the invariant property of action effects is not satisfied, e.g., when "moving" towards a barrier, the learned invariant action effect is moving forward, but the overall (actual) action effect is staying put. To bridge this discrepancy between the invariant action effect and the overall action effect, we introduce the individual action effect and the individual activation weight $\boldsymbol{\omega}$.

Individual Action Effect. We separate the action effect $A^{\phi}_{\mathcal{Z}}(\mathbf{z}, \mathbf{a})$ into two parts: the invariant one and the individual one. We let $A^{\phi}_{\mathcal{Z}^V}(\mathbf{z}, \mathbf{a})$ denote the invariant action effect and rewrite the *action effect similarity* $g(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_k, \mathbf{a}_k)$ in Eq (1) as

$$g(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_k, \mathbf{a}_k) = A^{\phi}_{\mathcal{Z}^V}(f^{\theta}_{\mathcal{S}}(\mathbf{s}_t), \mathbf{a}_t)^T A^{\phi}_{\mathcal{Z}^V}(f^{\theta}_{\mathcal{S}}(\mathbf{s}_k), \mathbf{a}_k)/\tau. \tag{3}$$

Then we let $A^{\phi}_{\mathcal{Z}^D}(\mathbf{z}, \mathbf{a})$ denote the individual action effect, which can be combined to attain the overall action effect $A^{\phi}_{\mathcal{Z}}(\mathbf{z}, \mathbf{a})$ as $A^{\phi}_{\mathcal{Z}}(\mathbf{z}, \mathbf{a}) = A^{\phi}_{\mathcal{Z}^V}(\mathbf{z}, \mathbf{a}) + A^{\phi}_{\mathcal{Z}^D}(\mathbf{z}, \mathbf{a})$. When the functions $A^{\phi}_{\mathcal{Z}^V}$ and $A^{\phi}_{\mathcal{Z}^D}$ are optimized by gradient descent, the tuple of the two action effects has a trivial solution where the invariant action effect can be zero and the individual action effect equals to the overall action effect. Specifically, suppose $A^{\phi^*}_{\mathcal{Z}}(\mathbf{z}, \mathbf{a})$ with parameter ϕ^* is the ground-truth overall action effect, the optimal invariant action effect should be $A^{\phi^*}_{\mathcal{Z}^V}(\mathbf{z}, \mathbf{a}_0) = \arg\min_{\mathbf{x}} \mathbb{E}_{(\mathbf{s},\mathbf{a})\sim\mathcal{D}, \mathbf{a}=\mathbf{a}_0} \|x - A^{\phi^*}_{\mathcal{Z}}(\mathbf{z}, \mathbf{a})\|_p$. The optimal individual action effect should be $A^{\phi^*}_{\mathcal{Z}^D}(\mathbf{z}, \mathbf{a}) = A^{\phi^*}_{\mathcal{Z}}(\mathbf{z}, \mathbf{a}) - A^{\phi^*}_{\mathcal{Z}^V}(\mathbf{z}, \mathbf{a})$. Since it is easy to get a local minimum at $\hat{\phi}$ satisfying $\left(A^{\hat{\phi}}_{\mathcal{Z}^V}(\mathbf{z}, \mathbf{a}), A^{\hat{\phi}}_{\mathcal{Z}^D}(\mathbf{z}, \mathbf{a})\right) = \left(\mathbf{0}, A^{\phi^*}_{\mathcal{Z}}(\mathbf{z}, \mathbf{a})\right)$, we are prone to learn a trivial solution that the invariant action effect collapse to a zero vector. To prevent this trivial collapse, we propose a normalization constraint on the individual action effect, i.e. the effects are restricted with L1 loss:

$$\mathcal{L}_{\text{norm-individual}}(\theta, \phi) = \mathbb{E}_{\mathbf{s}}[\|A^{\phi}_{\mathcal{Z}^D}(f^{\theta}_{\mathcal{S}}(\mathbf{s}), \mathbf{a})\|_1] \tag{4}$$

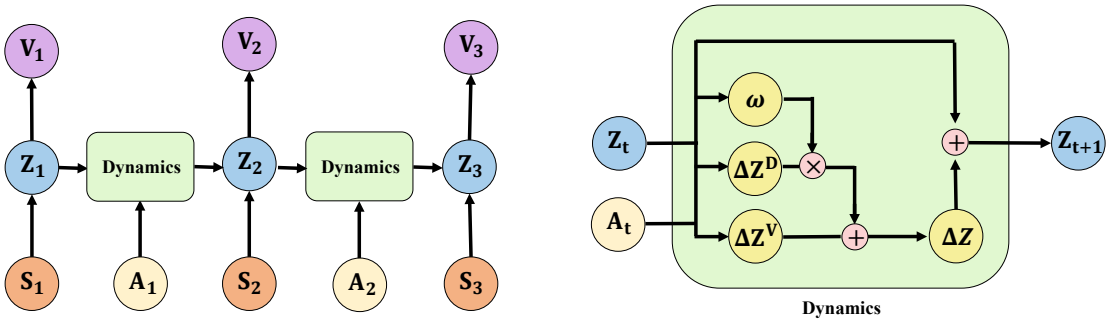


Figure 1: The architecture of IAEM. **Left**: the complete view of dynamics-based representation learning. $\mathbf{S}_t$ represents the observation, $\mathbf{A}_t$ represents the action, $\mathbf{Z}_t$ represents the latent representation and $\mathbf{V}_t$ represents the value of $\mathbf{S}_t$, which is the output of the Q-learning algorithms. **Right**: the details of the dynamics in the representation space. We learn three variables from the state representation and actions, including the invariant action effect $\Delta\mathbf{Z}^V$, the individual action effect $\Delta\mathbf{Z}^D$ and its weight $\boldsymbol{\omega}$. These variables are combined by the flow chart to compute the residual between two state representations $\Delta\mathbf{Z}$, where $\times$ represents dot product and $+$ represents addition.

With the gradient $\nabla\phi\mathbb{E}_{\mathbf{s}}[\|A^{\phi}_{\mathcal{Z}^D}(f^{\theta}_{\mathcal{S}}(\mathbf{s}),\mathbf{a})\|_1]$ from the loss (4), the gradient at $\hat{\phi}$ will not be zero, which prevents the trivial collapse and forces the model to reach the target solution ϕ^*.

Individual Activation Weight. Since the discrepancy only occurs in limited states in RL environments, those non-zero individual action effects should be activated in few states. However, without any constraint, non-zero individual action effects are abused in states, which breaks the invariance in action effects and leads to poor generalization ability. To solve this problem, we introduce a dynamic weight $\boldsymbol{\omega}(\mathbf{z},\mathbf{a}) \in [0,1]$ that determines when to activate the individual action effect by $A^{\phi}_{\mathcal{Z}}(\mathbf{z},\mathbf{a}) = A^{\phi}_{\mathcal{Z}^V}(\mathbf{z},\mathbf{a}) + \boldsymbol{\omega}(\mathbf{z},\mathbf{a})A^{\phi}_{\mathcal{Z}^D}(\mathbf{z},\mathbf{a})$. Though the normalization constraint on the individual action effect can also help alleviate the problem slightly, $\boldsymbol{\omega}(\mathbf{z},\mathbf{a})$ solves it by playing as a switch, which is much more direct and convenient. With the switch variable, we rewrite the transition loss as:

$$\begin{aligned}\mathcal{L}_{\text{transition}}(\theta,\phi) =&\mathbb{E}_{(\mathbf{s},\mathbf{a},\mathbf{s}')}[\mathbf{z} + A^{\phi}_{\mathcal{Z}^V}(\mathbf{z},\mathbf{a}) \\ &+ \boldsymbol{\omega}(\mathbf{z},\mathbf{a})A^{\phi}_{\mathcal{Z}^D}(\mathbf{z},\mathbf{a}) - \mathbf{z}']^2. \qquad (5)\end{aligned}$$

After defining all proposed new losses, we combine the two traditional losses $\mathcal{L}_{\text{reward}}$ and $\mathcal{L}_{\text{norm}}$ in preliminary with our proposed new losses to get the final optimization target of dynamics as:

$$\begin{aligned}\mathcal{L}_{\text{dynamics}}(\theta,\phi,\zeta) =&\mathcal{L}_{\text{invariant}}(\theta,\phi) + \mathcal{L}_{\text{transition}}(\theta,\phi) \\ &+ \mathcal{L}_{\text{reward}}(\theta,\zeta) + \mathcal{L}_{\text{norm}}(\theta) \\ &+ \mathcal{L}_{\text{norm-individual}}(\theta,\phi) \qquad (6)\end{aligned}$$

Notice that ϵ in $\mathcal{L}_{\text{norm}}$ is a hyperparameter that controls the scale of the embeddings and we set $\epsilon = 1$ in this paper, which follows previous works (van der Pol et al. 2020).

With these losses in the optimization target, the learned representations can hold the invariance property of action effects, which benefits sample efficiency and generalization ability. And the adaptation-related losses assure the invariance on even marginal conditions, which is the key challenge to practical RL environments.

Algorithm 1: Invariant Action Effect Model (IAEM)

Input: $\mathcal{D}$-empty replay buffer; θ,ϕ,ζ,ψ-initial network parameters, which are the encoder function, dynamics function, reward function and policy function respectively; N-the number of epochs.
for epoch $= 0$ to N **do**
  Get initial state $\mathbf{s}_0$
  for step $t = 0$ to terminal **do**
    Encode state $\mathbf{z}_t = f^{\phi}_{\mathcal{S}}(\mathbf{s}_t)$
    Execute action $\mathbf{a}_t = \pi(\mathbf{z}_t)$
    Record data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1}\}$
    Sample batch $\mathcal{B} \sim \mathcal{D}$
    Train dynamics, encoder and policy functions: $\Delta\phi,\Delta\theta,\Delta\zeta,\Delta\psi = \nabla\phi,\theta,\zeta,\psi(\mathcal{L}_{\text{all}})$ by Eq (8)
    Update parameters ϕ,θ,ζ,ψ with $\Delta\phi,\Delta\theta,\Delta\zeta,\Delta\psi$
  end for
end for

Training Algorithms

In this subsection, we first introduce the policy learning procedure and then provide the complete algorithm training procedure.

Policy Learning. The policy learning of IAEM could benefit straightforwardly from using any other existing variant of DQN (Hessel et al. 2018) or actor-critic architectures (Mnih et al. 2016). For fair and convenient comparison, we combine our representation learning approach with the DQN algorithm (Mnih et al. 2013) to devise a practical reinforcement learning method. Letting $Q^{\psi}_{\mathcal{Z}} : \mathcal{Z} \times \mathcal{A} \to \mathbb{R}$ de-

note the Q-value function in the representation space, and $Y = r + \gamma \max_{\mathbf{a}} Q_{\mathcal{Z}}^{\psi}(f_{\mathcal{S}}^{\theta}(\mathbf{s}'), \mathbf{a})$ denote the target value, the Q-value function training is done by minimizing:

$$\mathcal{L}_{\text{policy}}(\theta, \psi) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')}[Q_{\mathcal{Z}}^{\psi}(\mathbf{z}, \mathbf{a}) - Y]^2 \quad (7)$$

Notice that the gradient of loss (7) updates the parameters of both the encoder and the Q-value function. The policy π is derived from the learned value function Q as $\pi(\mathbf{s}) = \arg\max_{\mathbf{a}} Q_{\mathcal{Z}}^{\psi}(f_{\mathcal{S}}(\mathbf{s}), \mathbf{a})$.

Model Training. After defining the policy learning, we conclude the complete training process in IAEM. We train IAEM by updating three components: the Q-value function $Q_{\mathcal{Z}}^{\psi}$, the encoder $f_{\mathcal{S}}^{\theta}$ and the latent dynamics model $(P_{\mathcal{Z}}^{V}, P_{\mathcal{Z}}^{D}, \boldsymbol{\omega})$. At each training step, a sum of the aforementioned losses are minimized using gradient descent:

$$\mathcal{L}_{\text{all}}(\theta, \phi, \zeta, \psi) = \mathcal{L}_{\text{dynamics}}(\theta, \phi, \zeta) + \mathcal{L}_{\text{policy}}(\theta, \psi) \quad (8)$$

After each training step, the policy π is used to step in the environment, the data is collected in a replay buffer $\mathcal{D}$, and a batch is randomly selected to repeat training.

The detailed learning procedure is shown in Algorithm 1. Notice that our method could be combined with any RL algorithm in principle, including the policy gradient algorithms. Implementation details and hyperparameter values of IAEM are summarized in the appendix A.

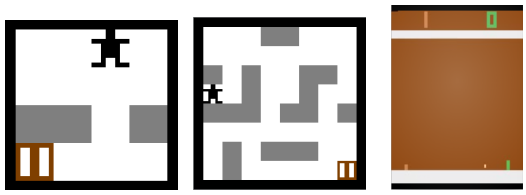


Figure 2: State observations for labyrinth task and Atari Games. **Left:** a simple task with 3×3 size. **Middle:** a difficult task with 8×8 size. In the labyrinth, the agent aims to bypass the grey wall and then reach the brown door. **Right:** an example of the Atari Games, the Pong. The goal of both two boards is to hit back the ball.

Experiments

We evaluate the *performance*, *sample efficiency*, and the *generalization ability* of IAEM on two widely-used benchmarks: Grid-World and Atari games. The Grid-World is a simple and typical environment that is convenient to visualize the learned representations and verify the advantages of the proposed model in few environment steps. The Atari Game is the most common benchmark to investigate the performance in pixel-based complex environments, where previous works provide many standard baseline algorithms. We evaluate our IAEM on two indexes: (1) Evaluating *sample-efficiency* by measuring how many steps it takes the best performing baselines to match IAEM performance in limited 20k (Grid-World) or 30M (Atari) steps and (2) Evaluating *performance* by measuring the ratio of the episode returns achieved by IAEM versus the best performing baselines at limited steps.

Baselines. We compare IAEM with two dynamics-based approaches (CRAR, PRAE) and one model-free algorithm (DQN): (1) DQN (Mnih et al. 2013) is the policy learning algorithm used in the two dynamics-based approaches, which can be seen as the blank control without any dynamics constraints. DQN takes visual observations as inputs directly and outputs the $Q(s, a)$ for policy learning. (2) PRAE (van der Pol et al. 2020) is a sota dynamics-based approach that builds an MDP homomorphism by keeping action-equivalence. PRAE learns a plannable MDP homomorphism with a dynamics-related constraint. (3) CRAR (François-Lavet et al. 2019) is another sota dynamics-based approach that captures the relation of the action effects in some toy environments by designing a Cosine-based similarity constraint.

Environment. *Grid-World*. We use a labyrinth MDP as a pixel-based Grid-World for convenient visualization of the representations, where the dynamic is a discrete state MDP. Fig. 2 shows their visual observation respectively and the following will give a detailed description. *Labyrinth Env.* The agent moves towards the four cardinal directions through four actions, except when the agent reaches the barriers (grey) or the wall (black), whose goal is to reach the door. In every step, the reward $r = 100$ if the agent reaches the door and $r = -1$ otherwise. The episode ends if the agent reaches the door or has taken more than 50 steps. The starting positions are selected randomly when a new episode starts. *Atari*. Compared to the Grid-World, Atari (Bellemare et al. 2012) is much more complex in both dynamics and observations. We experiment with IAEM in 8 different games of Atari.

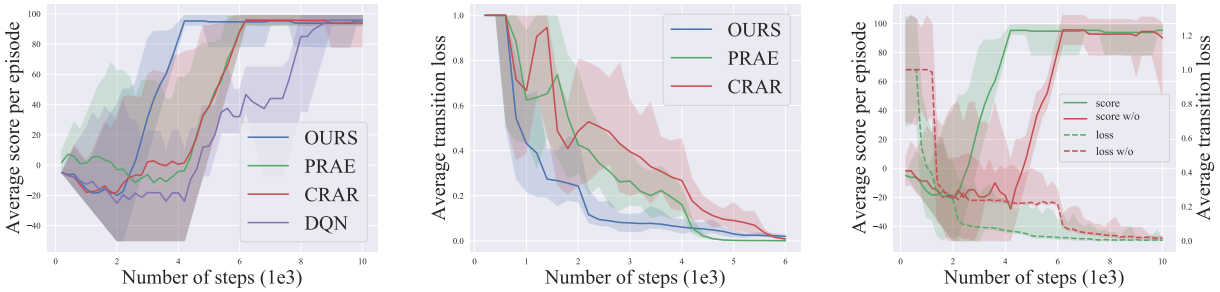
Implement Details. For the network architecture in Grid-World, we use the same network architecture as that in the two dynamics-based baselines. For the network architecture in Atari, we use a state-of-the-art DQN baseline *dopamine* (Castro et al. 2018). We directly make improvements based on *dopamine*. Following the preprocessing pipeline of previous works, we down-sample frames to 84×84 pixels, convert them to gray-scale, and stack 4 consecutive frames as one observation. More detailed architectural hyperparameters can be found in the appendix A.

Performance

The results in Fig. 3(a) show that our IAEM outperforms all three baselines by a large margin. Additionally, the results in Fig. 3(b) show the transition loss of three dynamics-based approaches, where the transition loss curves are significantly consistent with the performance curves. This consistency implies that a low transition loss helps to learn a good policy, which indicates that our IAEM achieves good performance by the well-learned latent dynamics. Moreover, in Atari Games, IAEM outperforms baselines by 1.4x gains averagely and achieves significant improvements on the majority (**75%**) in Fig. 4.

Sampling Efficiency

The results in Fig. 4 show that IAEM surpasses the best baseline on the sample efficiency in all eight games. In some games, we can use 65% to 80% fewer samples to reach the



(a) Comparison on averaged scores (b) Comparison on transition losses (c) The Ablation study.

Figure 3: Comparison of IAEM and baselines in Grid-World. **Left:** The comparison on averaged scores. IAEM can reach the optimal policy in 4000 time-steps while others need at least 6000 time-steps. **Middle:** The comparison on the transition loss. Optimization of the transition constraint in IAEM is faster than it in other baselines. **Right:** The ablation study on the invariant action effect. We deactivate the invariant action effect for ablation study, and the result shows that this component contributes to the IAEM sample efficiency.

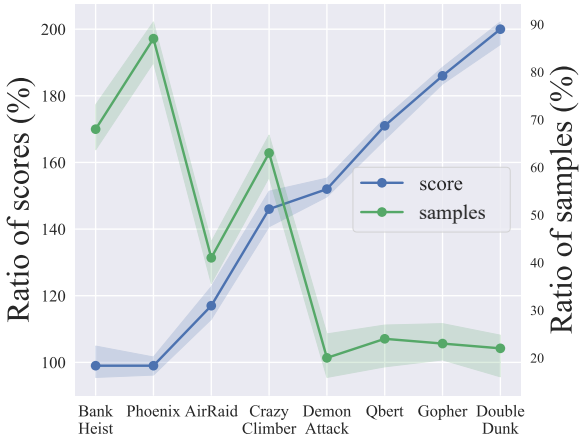


Figure 4: The comparison on scores and samples in Atari games. **Blue:** The ratio of IAEM to the best baseline on scores. **Green:** The ratio of IAEM to the best baseline on samples.

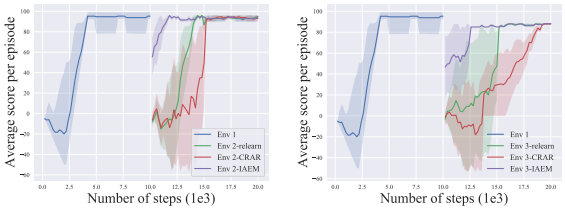


Figure 5: The generalization study for IAEM invariant action effect component. **Left:** A new environment of the same size but with different barriers. **Right:** A new environment of a larger size. The reported score is the mean of the running average of 3 independent runs.

same performances while in others the percents the ratio of saved samples is still more than 13%. Detailed results on sample efficiency can be found in appendix B.

IAEM outperforms those three baselines for the following reasons: (1) DQN learns policy directly from high-dimensional observations but IAEM learns from low-dimensional representations, which can alleviate the disturbance from irrelevant information; (2) PRAE learns representations without any prior knowledge about the action effect, which causes repeating learning in the same action effect among states, in contrast, IAEM saves samples by introducing the invariance relation in representations; (3) CRAR tries to introduce the relation of the action effect in the form of cos-based similarity, which suffers from the discrepancy problem and mainly works in toy environments. IAEM solves this problem by introducing the individual action effect and the switch variable ω.

Illustration of Generalization

The IAEM architecture has the advantage of explicitly training its different components, and hence can be used for generalization by retraining some of its components to adjust to new environments. In particular, one can enforce that states related to the same underlying dynamics but in different environments (e.g., a larger-size environment) are mapped into the representation space with a shared invariant action effect. In that case, an agent trained in a small environment can be deployed in a complex and large setting with limited training.

To illustrate the possibility of using the IAEM for generalization, we consider putting the agent into the two settings after pretraining: (1) environments with the same size but barriers in different positions. (2) environments with a larger size and more barriers, such as the middle of Fig. 2. In the second set, the env-1 experience available to the agent is converted to the Env-3 version (the larger version) by padding the blank space. On the first 1e4 steps, training is done on the original environment (Env-1) while for the remaining 1e4 steps, training is done on the new environments

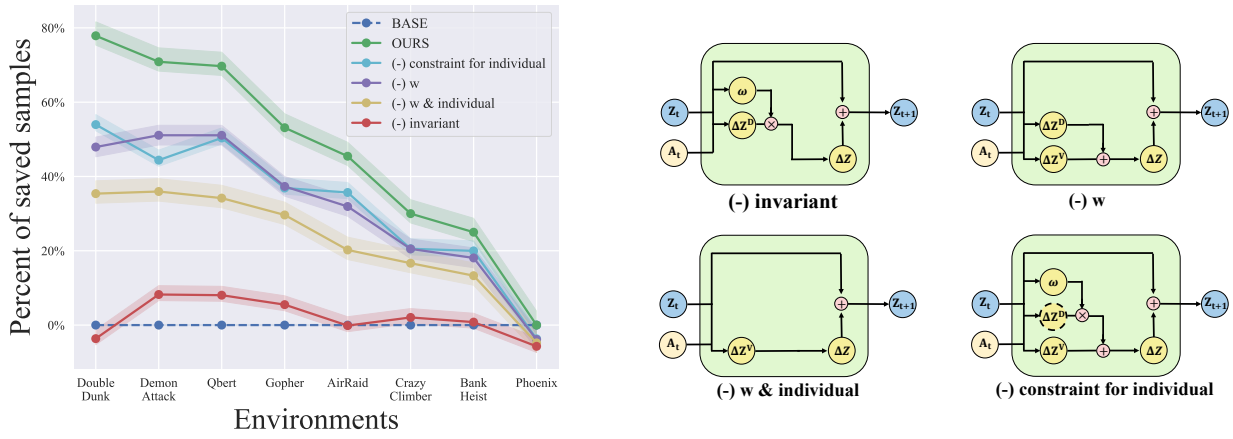


Figure 6: **Left:** The ablation study for IAEM on sample-efficiency in various Atari games. **Right:** The architectures of the dynamics deactivating some components. Results show that ω, the individual action effect and the invariant action effect all contribute to the IAEM sample-efficiency.

(Env-2 and Env-3).

Fig. 5 shows that, with the generalization procedure, little retraining is sufficient in contrast to directly learning. We take CRAR (François-Lavet et al. 2019) as the baseline since it also claims the generalization ability with only replacing some components. IAEM outperforms CRAR because CRAR can only generalize in environments with different rendering but doesn't learn a generalizable action effect w.r.t a fixed dynamics mechanism.

Visualization of Learned Representations

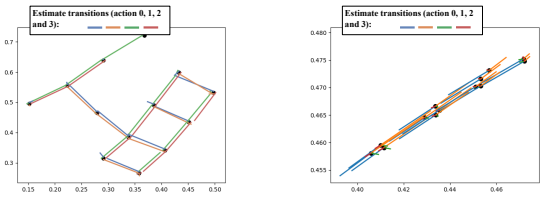


Figure 7: Learned abstract states and actions of two different approaches in the labyrinth task. **Left:** IAEM. **Right:** PRAE. The two learned representations are trained with the same steps.

To understand the benefits from the invariant action effect, we visualize the learned representations in Fig. 7, where vertex represent the state representations $\mathbf{z}$, edges represent the action effects $A^{\phi}_{\mathcal{Z}}(\mathbf{z}, \mathbf{a})$ and colors represent the action classes. The representations in the two figures are trained with the same steps in the same environment. Results show that: (1) The directions of edges are clustered by their colors, which verifies that we get invariant action effects in the representation space by the contrastive-based loss. Additionally, the learned action effect can be easily reused in other states since they are similar. (2) The vertex are connected concretely by the edges, which means that we get concise dynamics in the representation space and verifies that the individual action effects work on making up the discrepancy. In the perspective of disentanglement, the learned representation disentangles the overall action effect by splitting it into the invariant one and the individual one. In contrast, in PRAE the learned representations are less interpretable and the action effects cannot be reused.

Ablation Study

To evaluate the contribution of each component, we perform an ablation study for IAEM. The results in Fig. 3(c) show that the invariant component contributes to the optimization of both the transition loss and the policy loss. Furthermore, in Fig. 6 we deactivate more components in Atari, including the invariant action effect, the switch variable ω, the normalization constraint for the individual action effect, and the whole individual action effect. Results show that all these components contribute to the IAEM sample efficiency, where the invariant action effect plays the most important role in IAEM.

Conclusion

In this paper, we propose an effective approach IAEM to address the problems of sample efficiency and generalization in dynamics-based approaches of learning abstract representations. The main idea of IAEM is to capture the invariance relations between action effects, through which we learn invariant action effects and build an architecture to use this effect in practice. Extensive experiments on the Grid-World and Atari demonstrate that our IAEM outperforms other state-of-the-art dynamics-based approaches, saves samples, and has great generalization ability. Moreover, IAEM can be adapted into more RL algorithms, including policy gradient algorithms and model-based approaches.

References

Anand, A.; Racah, E.; Ozair, S.; Bengio, Y.; Côté, M.; and Hjelm, R. D. 2019. Unsupervised State Representation Learning in Atari. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 8766–8779.

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2012. The Arcade Learning Environment: An Evaluation Platform for General Agents. *CoRR*, abs/1207.4708.

Bellman, R. 1957. A Markovian decision process. *Journal of mathematics and mechanics*, 6(5): 679–684.

Biza, O.; Jr., R. P.; van de Meent, J.; and Wong, L. L. S. 2020. Learning discrete state abstractions with deep variational inference. *CoRR*, abs/2003.04300.

Castro, P. S.; Moitra, S.; Gelada, C.; Kumar, S.; and Bellemare, M. G. 2018. Dopamine: A Research Framework for Deep Reinforcement Learning. *CoRR*, abs/1812.06110.

Corneil, D. S.; Gerstner, W.; and Brea, J. 2018. Efficient Model-Based Deep Reinforcement Learning with Variational State Tabulation. *CoRR*, abs/1802.04325.

Dean, T. L.; and Givan, R. 1997a. Model Minimization in Markov Decision Processes. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Prodence, Rhode Island, USA*, 106–111.

Dean, T. L.; and Givan, R. 1997b. Model Minimization in Markov Decision Processes. In Kuipers, B.; and Webber, B. L., eds., *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference, AAAI 97, IAAI 97, July 27-31, 1997, Providence, Rhode Island, USA*, 106–111. AAAI Press / The MIT Press.

Dwibedi, D.; Tompson, J.; Lynch, C.; and Sermanet, P. 2018. Learning Actionable Representations from Visual Observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2018, Madrid, Spain, October 1-5, 2018*, 1577–1584.

Feng, H.; You, Z.; Chen, M.; Zhang, T.; Zhu, M.; Wu, F.; Wu, C.; and Chen, W. 2021a. KD3A: Unsupervised Multi-Source Decentralized Domain Adaptation via Knowledge Distillation. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 3274–3283. PMLR.

Feng, H.-Z.; Kong, K.; Chen, M.; Zhang, T.; Zhu, M.; and Chen, W. 2021b. SHOT-VAE: Semi-supervised Deep Generative Models With Label-aware ELBO Approximations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8): 7413–7421.

François-Lavet; Bengio, Y.; Precup, D.; and Pineau, J. 2019. Combined Reinforcement Learning a Abstract Representations. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 3582–3589.

Gelada, C.; Kumar, S.; Buckman, J.; Nachum, O.; and Bellemare, M. G. 2019. DeepMDP: Learning Continuous Latent Space Models for Representation Learning. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2170–2179.

Hafner, D.; Lillicrap, T. P.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019. Learning Latent Dynamics for Planning from Pixels. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, 2555–2565.

Hénaff, O. J.; Srinivas, A.; Fauw, J. D.; Raza, A.; Doersch, C.; Eslami, S. M. A.; and van den Oord, A. 2019. Data-Efficient Image Recognition with Contrastive Predictive Coding. *CoRR*, abs/1905.09272.

Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M. G.; and Silver, D. 2018. Rainbow: Combining Improvements in Deep Reinforcement Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 3215–3222.

Higgins, I.; Pal, A.; Rusu, A. A.; Matthey, L.; Burgess, C.; Pritzel, A.; Botnick, M.; Blundell, C.; and Lerchner, A. 2017. DARLA: Improng Zero-Shot Transfer in Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning*, 1480–1490.

Igl, M.; Zintgraf, L. M.; Le, T. A.; Wood, F.; and Whiteson, S. 2018. Deep Variational Reinforcement Learning for POMDPs. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 2122–2131.

Kipf, T. N.; van der Pol, E.; and Welling, M. 2020. Contrastive Learning of Structured World Models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 1928–1937.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. A. 2013. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nat.*, 518(7540): 529–533.

Randran, B.; and Barto, A. G. 2001. Symmetries and model minimization in markov decision processes. *CoRR*.

Randran, B.; and Barto, A. G. 2004. Approximate homomorphisms: A framework for non-exact minimization in Markov decision processes. *CoRR*.

Sermanet, P.; Lynch, C.; Chebotar, Y.; Hsu, J.; Jang, E.; Schaal, S.; and Levine, S. 2018. Time-Contrastive Networks: Self-Supervised Learning from Video. In *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, 1134–1141.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nat.*, 529(7587): 484–489.

Srinivas, A.; Laskin, M.; and Abbeel, P. 2020. CURL: Contrastive Unsupersed Representations for Reinforcement Learning. *CoRR*, abs/2004.04136.

Tassa, Y.; Doron, Y.; Muldal, A.; Erez, T.; Li, Y.; de Las Casas, D.; Budden, D.; Abdolmaleki, A.; Merel, J.; Lefrancq, A.; Lillicrap, T. P.; and Riedmiller, M. A. 2018. DeepMind Control Suite. *CoRR*, abs/1801.00690.

van den Oord, A.; Li, Y.; and nyals, O. 2018. Representation Learning with Contrastive Predictive Coding. *CoRR*, abs/1807.03748.

van der Pol, E.; Kipf, T.; Oliehoek, F. A.; and Welling, M. 2020. Plannable Approximations to MDP Homomorphisms: Equivariance under Actions. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, 1431–1439.

Zhang, A.; Lyle, C.; Sodhani, S.; Filos, A.; Kwiatkowska, M.; Pineau, J.; Gal, Y.; and Precup, D. 2020a. Invariant Causal Prediction for Block MDPs. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, 11214–11224.

Zhang, A.; McAllister, R.; Calandra, R.; Gal, Y.; and Levine, S. 2020b. Learning Invariant Representations for Reinforcement Learning without Reconstruction. *CoRR*, abs/2006.10742.

Zhang, T.; Feng, H.; Chen, W.; Chen, Z.; Zheng, W.; Luo, X.-N.; Huang, W.; and Tung, A. K. H. 2021. ChartNavigator: An Interactive Pattern Identification and Annotation Framework for Charts. *IEEE Transactions on Knowledge and Data Engineering*, 1–1.

Experimental Details

Grid-World

The network architecture

For the labyrinth task, the state abstraction function consists of 3 convolutional layers followed by 2 fully connected layers (Table 1); The action abstraction function consists of 3 fully connected layers(Table 2); The reward abstraction function consists of 3 fully connected layers(Table 3)

state abstraction					
	Convolutional layers				
	in size	out size	kernel	stride	padding
layer 1	3	32	4	2	1
active	LeakyReLU				
layer 2	32	64	4	2	1
active	LeakyReLU				
layer 3	64	64	4	2	1
active	LeakyReLU				
	fully connection layers				
	in size	out size			
layer 1	1024	256			
active	LeakyReLU				
layer 2	256	latent size			
active	Sigmoid				

Table 1: The details of the state abstraction network.

action abstraction					
	fully connection layers				
	in size	out size			
layer 1	latent size	256			
active	LeakyReLU				
layer 2	256	256			
active	LeakyReLU				
layer 3	256	latent size*action shape			
active	Sigmoid				

Table 2: The details of the action abstraction network.

reward function					
	fully connection layers				
	in size	out size			
layer 1	latent size	256			
active	LeakyReLU				
layer 2	256	256			
active	LeakyReLU				
layer 3	256	action shape			
active	Sigmoid				

Table 3: The details of the reward function network.

The Hyperparameters

The hyperparameters of the baselines and our approach in the experiments (Table 4).

	Hyperparameters	
Commen	γ	0.99
	batch size	128
	learning rate	0.0001
	max norm of gradients	0.5
	step size	200
	buffer size	2000
	iterations per epoch	4
	target network interval(iteration)	1
	number of epochs	40
IAEM	τ	0.1
	ϵ	1

Table 4: The details of the hyperparameters in Grid-World.

Atari Games

We implement baselines and IAEM in Atari games based on *Dopamine*, which is a research framework for fast prototyping of reinforcement learning algorithms. The hyperparameters of the baselines and our approach in the experiments (Table 4).

	Hyperparameters	
Commen	γ	0.99
	batch size	32
	learning rate	0.00025
	max norm of gradients	0.5
	step size	250000
	buffer size	1000000
	target network interval(steps)	8000
IAEM	τ	0.1
	ϵ	1

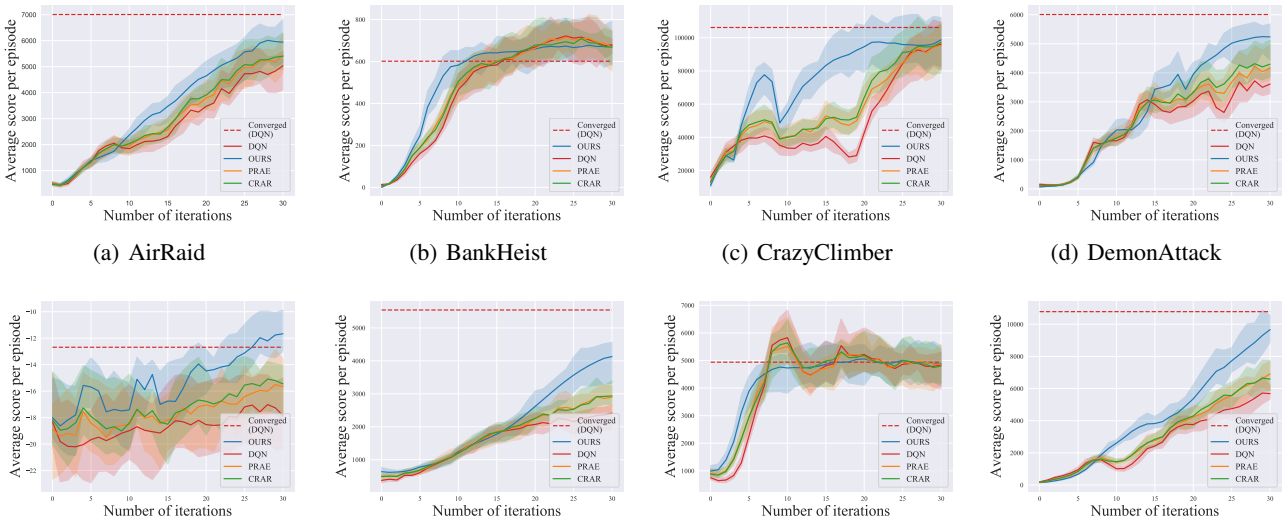
Table 5: The details of the hyperparameters in Atari games.

Experimental Results

We show the full results of eight Atari games (see Figure 8) and also make a comparison on samples (see Figure 9). The statistical result is shown in Table 6.

Table 6: Samples for converged performance needed by our IAEM and the best baseline on Atari games. RATIO represents the ratio of IAEM compared with the baseline.

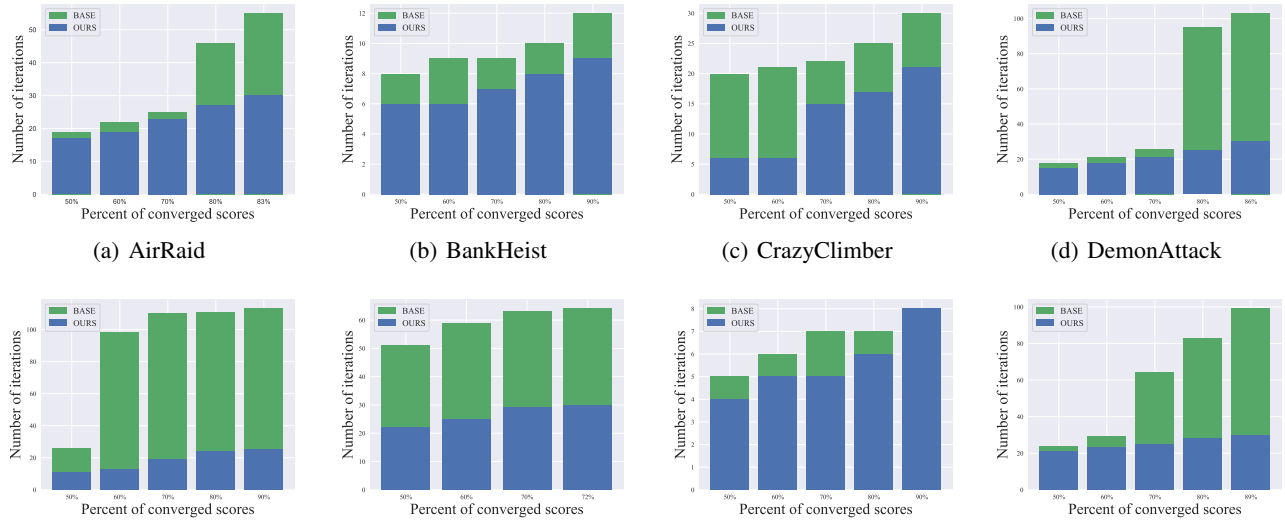
GAME	OURS	BASELINE	RATIO
AIRRAID	26	43	60%
BANKHEIST	13	19	68%
CRAZYCLIMBER	21	27	77%
DEMONATTACK	25	96	26%
DOUBLEDUNK	28	134	20%
GOPHER	29	62	46%
PHOENIX	7	8	87%
QBERT	30	84	35%
MEAN	-	-	47%



(a) AirRaid (b) BankHeist (c) CrazyClimber (d) DemonAttack

(e) DoubleDunk (f) Gopher (g) Phoenix (h) Qbert

Figure 8: Comparison of IAEM and the best baseline in eight Atari games. Every iteration, one million steps are taken using the learned policy. The reported score is the mean of that running average (3 independent runs). Dotted lines represent asymptotic scores after 200 iterations according to a public resource (Castro et al. 2018)



(a) AirRaid (b) BankHeist (c) CrazyClimber (d) DemonAttack

(e) DoubleDunk (f) Gopher (g) Phoenix (h) Qbert

Figure 9: Comparison of IAEM and the best baseline in eight Atari games. Every bar represents how many iterations are necessary to achieve different percentages of the converged performance.