Towards A Unified Policy Abstraction Theory and Representation Learning Approach in Markov Decision Processes

Min Zhang
College of Intelligence and Computing, Tianjin University
Tianjin, China
min_zhang@tju.edu.cn

Hongyao Tang*
College of Intelligence and Computing, Tianjin University
Tianjin, China
MoE Key Laboratory of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China
Hefei, China
tanghongyao@tju.edu.cn

Jianye Hao
College of Intelligence and Computing, Tianjin University
Tianjin, China
jianye.hao@tju.edu.cn

Yan Zheng
College of Intelligence and Computing, Tianjin University
Tianjin, China
yanzheng@tju.edu.cn

ABSTRACT

In intelligent decision-making systems, how policy is represented and optimized is a fundamental problem. The root challenge stems from the large scale and the high complexity of policy space. Towards a desirable surrogate policy space, recent policy representations in a low-dimensional latent space has revealed its potential in improving both evaluation and optimization of policy. The key question to answer in this line of research is *by what criterion the policy space should be abstracted* for favorable compression and generalization. However, both the theory of policy abstraction and the method of policy representation learning are under-studied. In this work, we first make efforts to fill the vacancy. First, we propose a unified policy abstraction theory, containing three types of policy abstraction and explaining their partial ordering relationship. Then, we generalize policy abstractions to three policy metrics that quantify the distance between policies. Further, we propose a policy representation learning approach and policy optimization algorithm based on deep metric learning. Our study highlights the importance of policy abstraction theory and representation method, demonstrating their effectiveness in compressing policy space, characterizing policy differences, and conveying policy generalization.

KEYWORDS

Policy Space Compression; Policy Representation

ACM Reference Format:
Min Zhang, Hongyao Tang, Jianye Hao, and Yan Zheng. 2026. Towards A Unified Policy Abstraction Theory and Representation Learning Approach in Markov Decision Processes. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 13 pages. https://doi.org/10.65109/IRYM6625

*Corresponding author: tanghongyao@tju.edu.cn.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/IRYM6625

1 INTRODUCTION

How to obtain the optimal policy is the ultimate problem in decision-making systems, such as Game Playing [15], Robotics Manipulation [26], Natural Language Processing [20]. To address the problem, a lot of work carry out studies on policy with different focal points, e.g., how policy can be well represented [14, 30], how to optimize policy [11, 25] and how to analyze and understand agents' behaviors [9, 31].

The root challenge to the studies on policy is the large scale and the high complexity of policy space, especially in real-world scenarios where the difficulty of policy learning is escalated severely [16]. Intuitively, the challenge can be addressed if we have an ideal surrogate policy space, which is compact in scale while keeping crucial features of the policy space. Related to this idea, some recent works study low-dimensional latent representations of policies, which plays a significant role in Reinforcement Learning (RL) [28], Opponent Modeling [8], Policy Adaptation [22, 24], Behavioral Characterization [13], etc. In these works, most of policy representations [4, 10, 16, 19, 28] extract the information of policy by encoding policy's state-action pairs (or trajectories) and optimizing a policy recovery objective [8, 22, 28]. Rather than policy distribution, some other works learn policy embedding using the state visitation distribution induced by the policy or policy network parameters [13, 28].

Despite notable progress in these studies, the key question of *by what criterion the policy space should be abstracted* for favorable compression and generalization remains essentially unaddressed. In other words, existing research has not yet established a systematic theoretical framework for policy abstraction, leaving researchers without effective guidance when selecting an appropriate policy abstraction for specific downstream tasks (such as the RL [28], Opponent Modeling [8] mentioned above). For instance, it is currently unclear whether there are other effective ways for compressing policy space besides the state-action distribution and state visitation distribution of the policy; what the inherent connections are between different methods; and what the essential differences are in policy space compressed using different methods.

Table 1: Properties of different policy abstractions, including the additional ones introduced in Appendix A.1. We use (+) and (−) to denote the higher or lower degree at the same level.

Abstraction	Abstraction Criterion (for $\pi_1, \pi_2, \forall s, s', a \in S^2 \times A$)	Fineness	Task Relevance
f_Θ	Policy Parameter Equivalence ($\theta_1 = \theta_2$)	Highest	None
f_π	Action Distribution Equivalence ($\pi_i(a \mid s) = \pi_j(a \mid s)$)	High	Low
f_{P^π}	Dynamics Influence Equivalence ($P^{\pi_i}(s'\|s) = P^{\pi_j}(s'\|s)$)	Middle (+)	Middle (−)
$f_{d_s^\pi}$	Dynamics Influence Equivalence ($d_s^{\pi_i}(s') = d_s^{\pi_j}(s')$)	Middle	Middle
f_{d^π}	Dynamics Influence Equivalence ($d^{\pi_i}(s) = d^{\pi_j}(s)$)	Middle (−)	Middle (+)
f_{V^π}	Value Function Equivalence ($V^{\pi_i}(s) = V^{\pi_j}(s)$)	Low	High
f_{J^π}	Value Function Equivalence ($\mathbb{E}_{s_0\sim\rho_0}[V^{\pi_i}(s_0)] = \mathbb{E}_{s_0\sim\rho_0}[V^{\pi_j}(s_0)]$)	Low (−)	High (+)
f_0	Triviality (taking all policies as the same)	Lowest	None

To this end, our work makes pioneering explorations in both theory and method to fill this research gap. First, we propose a unified theory of policy abstraction. We start from proposing three types of policy abstraction: *distribution-irrelevance*, *influence-irrelevance*, and *value-irrelevance*. The abstractions follow different criteria, each of which is associated with unique aspects of policy and is based on corresponding policy equivalence relations. Further, we generalize the exact equivalence relations to policy metrics, allowing quantitatively measuring the distance (i.e., similarity) between policies. Finally, we introduce a policy representation learning approach based on policy metrics. We propose the *alignment loss* as a unified objective function of policy representation learning. It encourages policy representation to render specific abstraction criteria by minimizing the difference between the distance of policy embeddings and the quantity measured by the policy metrics. In addition, we conduct various experiments to better elucidate the effects of different policy abstractions, metrics, and representations. First, we make use of multiple representative finite MDPs to demonstrate how the original policy space is compressed differently by our proposed abstractions. Then, we perform a quantitative policy comparison with our derived policy metrics. Finally, we experimentally validate that integrating the proposed policy representation into the typical RL algorithm PPO [25] can significantly improve its performance.

Our systematic experimental investigations provide profound insights into the critical role of policy abstraction in RL: (1) equivalent or approximately equivalent reductions on policy sets or spaces can be achieved; (2) qualitative and quantitative comparisons of pairwise policies can be achieved; (3) the construction of policy latent space representations enables the approximation and generalization of policy-conditioned objective functions and improves policy optimization and search processes. Our contributions can be summarized as follows:

- We are the first to propose a unified theory of policy abstraction, upon which we further develop a novel policy representation learning method.
- We perform systematic experiments to demonstrate the effects of different policy abstractions, metrics, and representations.
- We pave the way for leveraging policy abstractions to reduce the difficulty of policy learning in large-scale and highly complex policy spaces.

2 BACKGROUND

Reinforcement Learning. We consider a Markov Decision Process (MDP) [21] typically defined by a five-tuple $\langle S, A, P, R, \gamma \rangle$, with the state space S, the action space A, the transition probability $P : S \times A \to \Delta(S)$, the reward function $R : S \times A \to \mathbb{R}$ and the discount factor $\gamma \in [0, 1)$. $\Delta(X)$ denotes the probability distribution over X. A stationary policy $\pi : S \to \Delta(A)$ defines how the agent behaves under specific states. An agent interacts with the MDP at discrete timesteps by its policy π, generating trajectories with $s_0 \sim \rho_0(\cdot)$, $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ and $r_t = R(s_t, a_t)$, where ρ_0 is the initial state distribution. We use $P^\pi(s'|s) = \mathbb{E}_{a\sim\pi(\cdot|s)}P(s'|s, a)$ to denote the distribution of next state s' when performing policy π at state s. For a policy π, the return $G_t = \sum_{t=0}^{\infty} \gamma^t r_t$ is the random variable for the sum of discounted rewards while following π, whose distribution is denoted by Z^π. The value function of policy π defines the expected return for state s, i.e., $V^\pi(s) = \mathbb{E}_\pi[G_t \mid s_0 = s]$. The goal of an RL agent is to learn an optimal policy π^* that maximizes $J(\pi) = \mathbb{E}_{s_0\sim\rho_0(\cdot)}[V^\pi(s_0)]$.

Metric Learning. A metric d is used to quantify the *distance* between two data elements in a general sense. We recall the standard definitions of both metric and pseudo-metric in Appendix B. In this paper, we will use *metric* to stand for *pseudo-metric* for brevity. Typically metric learning aims to reduce the distance between similar data and increase the distance between dissimilar data. With nonlinear transformation offered by deep neural networks, Deep Metric Learning allows us to find such optimal metrics by optimizing a latent representation space of raw data.

The position of our work. For RL tasks, [28] utilizes action distribution reconstruction and contrast learning principles to learn policy representation. [4], [10] adopts policy fingerprints as differentiable policy representations obtained by concatenating the action distribution of the policy in a set of key states. [5] directly compresses the policy parameters into a vector, regarding it as a form of policy representation. For policy adaptation, [22, 24] utilizes action distribution reconstruction to learn policy representation. In a multi-agent system, [8] converts opponent modeling into an opponent policy representation learning problem and attempts to distinguish opponents by their policy representations. In addition, [16] investigates policy space compression by formulating a Set Covering problem with Rényi Divergence of the discounted state-action distribution of policies as a metric. [19] defines a trajectories-based Behavioral Embedding Map (BEM) to measure the similarity among

Table 2: A overview of policy representation-related work: original and promising policy abstraction criteria for corresponding downstream tasks based on our policy abstraction theory.

Tasks	Prior Policy Representation	Original Abstraction Criterion	Promising Abstraction Criterion
Reinforcement Learning	Vectorized Network Parameters [5]	Policy Parameter Equivalence	f_{V^π}
	Contrastive OPR/SPR [28]	Policy Instance Contrast	f_{V^π}
	Policy Recovery OPR/SPR [28]	Action Distribution Equivalence	f_{V^π}
Policy Adaptation	Network Fingerprint [4, 10]	Action Distribution Equivalence	f_{V^π}
	Policy Recovery [22, 24]	Action Distribution Equivalence	f_{P^π},f_{V^π}
Opponent Modeling	Generative Representation [8]	Action Distribution Equivalence	f_{P^π},f_{V^π}
	Discriminative Representation [8]	Policy Instance Contrast	f_{P^π},f_{V^π}
Policy Space Compression	α-compression [16]	Action Distribution Equivalence	f_{P^π},f_{V^π}
Policy Search	Behavior Embedding [19]	Trajectory Similarity	f_π,f_{P^π},f_{V^π}

policies for guiding policy optimization. While existing works have demonstrated the application value of policy representation in downstream tasks such as RL and policy adaptation, they have not yet established a systematic theoretical framework for policy abstraction. Our work fills this gap and provides effective guidance for selecting appropriate policy abstraction for specific downstream tasks.

3 POLICY ABSTRACTION THEORY

In this section, we propose a unified policy abstraction theory. First, we formally define policy abstraction and then propose three types of policy abstraction. Finally, we analyze the core properties of policy abstraction.

3.1 Policy Abstraction

Following the classic definition of an abstraction [6], we first propose a general definition of policy abstraction as follows:

Definition 1 (Policy Abstraction)**.** A policy abstraction $f : \Pi \to \mathcal{X}$, is a mapping from ground policy space Π to an abstract space $\mathcal{X}$. $f(\pi) \in \mathcal{X}$ represents the abstract policy corresponding to a ground policy $\pi \in \Pi$. The inverse image of $f^{-1}(\chi)$ with $\chi \in \mathcal{X}$ is defined as the set of all ground policies corresponding to χ under the abstraction function f.

It is apparent that there are many such abstractions since we may have many possible ways to partition the policy space. However, we are only interested in some useful ones among them that follow specific abstraction criteria to preserve the essential features related to decision-making. In this paper, we propose the following three types of policy abstraction:

Definition 2. Given an MDP and a ground policy space Π, for any two policies $\pi_i, \pi_j \in \Pi$, we define three types of policy abstraction as follows:

1. A distribution-irrelevance abstraction (f_π) is such that for all $s \in S$, $a \in A$, $f_\pi(\pi_i) = f_\pi(\pi_j)$ implies that $\pi_i(a \mid s) = \pi_j(a \mid s)$.
2. An influence-irrelevance abstraction (f_{P^π}) is such that for all $s, s' \in S$, $f_{P^\pi}(\pi_i) = f_{P^\pi}(\pi_j)$ implies that $P^{\pi_i}(s'|s) = P^{\pi_j}(s'|s)$.
3. A value-irrelevance abstraction (f_{V^π}) is such that for all $s \in S$, $f_{V^\pi}(\pi_i) = f_{V^\pi}(\pi_j)$ implies that $V^{\pi_i}(s) = V^{\pi_j}(s)$.

These abstractions aggregate policies based on the corresponding equivalence relations with respective concerns on different features of policy. Intuitively, f_π preserves the action distribution of the policy; f_{P^π} preserves the state transition distribution induced by the policy, i.e., the influence caused by the policy on the environment; and f_{V^π} preserves the value function of the policy. In addition to the policy abstractions introduced in Definition 2, we provide some other ones in Appendix A.1. Refer to Tab. 1 for the complete criterion which outlines that f_{P^π},$f_{d_s^\pi}$, and f_{d^π} belong to the influence-irrelevance abstraction type, while f_{V^π} and f_{J^π} belong to the value-irrelevance abstraction type.

3.2 Properties of Policy Abstraction

Superficially, the three abstractions proposed preserve features that are progressively more relevant to decision-making in the learning task (called task relevance), but essentially, what is the relationship between the three abstractions? To investigate the problem, we define the *fineness* of policy abstractions to prove how the three abstractions are related.

Definition 3 (Abstraction Fineness)**.** Let F_Π denote the set of abstractions on ground policy space Π. Suppose $f_1, f_2 \in F_\Pi$. We say f_1 is finer than f_2, denoted $f_1 \succeq f_2$, $iff \; \forall \; \pi_1, \pi_2 \in \Pi$, $f_1(\pi_1) = f_1(\pi_2)$ implies $f_2(\pi_1) = f_2(\pi_2)$. If, $f_1 \neq f_2$, then f_1 is strictly finer than f_2, denoted $f_1 \succ f_2$. In contrast, we may also say f_2 is (strictly) coarser than f_1, denoted $f_2 \preceq f_1$ ($f_2 \prec f_1$).

The relation $\succeq$ is a partial ordering since it satisfies self-reflexivity, antisymmetry, and transitivity. Consider the set of possible policy abstractions, while the coarsest abstraction (f_0) is the trivial representation where all policies are treated as the same; while the finest abstraction is the identity representation, e.g., $f_\Theta(\pi_\theta) = \theta$ for a policy neural network parameterized with $\theta \in \Theta$. With the partial ordering $\succeq$, we further derive the following theory.

Theorem 1 (Partial Ordering ($\succeq$))**.** Under the Definition 2 and 3, if the reward function R depends only on state $s \in S$, we have $(f_\Theta \succeq) f_\pi \succeq f_{P^\pi} \succeq f_{V^\pi} (\succeq f_0)$.

Proof. We prove the partial ordering ($\succeq$) of the Theorem 1 one by one in the following.

❶ $f_\pi \succeq f_{P^\pi}$. Given an MDP M, two policies $\pi_i, \pi_j \in \Pi$. We define P as the transition probability and use $P^\pi(s'|s) = \mathbb{E}_{a\sim\pi(\cdot|s)} P(s'|s,a)$ to denote the distribution of next state s' when performing policy π at state s, respectively. Then, we have

$$\begin{aligned} P^{\pi_i}(s'|s) &= \mathbb{E}_{a\sim\pi_i(\cdot|s)} P(s'|s,a), \\ P^{\pi_j}(s'|s) &= \mathbb{E}_{a\sim\pi_j(\cdot|s)} P(s'|s,a). \end{aligned} \tag{1}$$

If $f_\pi(\pi_i) = f_\pi(\pi_j)$, with the Definition 2, we have $\forall s \in S, \forall a \in A$, $\pi_i(a|s) = \pi_j(a|s)$. Combined with Eq. 1, we derive,

$$\forall s, s' \in S, P^{\pi_i}(s'|s) = P^{\pi_j}(s'|s). \tag{2}$$

Recall the definition of f_{P^π}, we can obtain $f_\pi \succeq f_{P^\pi}$.

❷ $f_{P^\pi} \succeq f_{V^\pi}$. Given an MDP M, two policies $\pi_i, \pi_j \in \Pi$, a reward function R. Start with the definition of the value function $V^\pi(\cdot)$, we consider the two cases:

Case 1. The reward function R depends only on state $s \in S$, we derive the value function:

$$\forall s \in S, V^\pi(s) = R(s) + \sum_{s'} P^\pi(s' \mid s) V^\pi(s'). \tag{3}$$

If $f_{P^\pi}(\pi_i) = f_{P^\pi}(\pi_j)$, with the Definition 2, we have $\forall s, s' \in S, P^{\pi_i}(s'|s) = P^{\pi_j}(s'|s)$. When the Eq. 3 holds, we have:

$$\forall s \in S, V^{\pi_i}(s) = V^{\pi_j}(s). \tag{4}$$

Recall the definition of f_{V^π}, we can obtain $f_{P^\pi} \succeq f_{V^\pi}$.

Case 2. The reward function R depends on both state $s \in S$ and action $a \in A$, we derive the value function:

$$V^\pi(s) = \sum_a \pi(a \mid s) \left(R(s,a) + \sum_{s'} P(s' \mid s, a) V^\pi(s') \right). \tag{5}$$

If $f_{P^\pi}(\pi_i) = f_{P^\pi}(\pi_j)$, with the Definition 2, we have $\forall s, s' \in S, P^{\pi_i}(s' \mid s) = P^{\pi_j}(s' \mid s)$. Unlike the Eq.3, π_i may not be equivalent to π_j regarding the abstraction criterion of value irrelevance. Therefore, the partial ordering ($f_{P^\pi} \succeq f_{V^\pi}$) is not obtained under Case 2. Nevertheless, Case 1 is fairly standard across a broad set of real-world RL problems.

Remark 1 (More discussions on $R(s,a)$ and $R(s)$): For Case 2 discussed in the proof of Theorem 1, i.e., the reward function R depends on both state $s \in S$ and action $a \in A$, we can further separate it into two categories according to our knowledge on real-world decision-making problems:

- In our first category, the dependence of R on state and action is due to the consequence of s, a in leading the decision system into the specific new state s', i.e., $R(s,a) = \sum_{s'} P(s'|s,a)U(s')$ where $U(s')$ is the utility of s' (we adopt the expectation form for the convenience of discussion). In the cases that fall into this category, it can be easy to re-define the reward function by the utility function, i.e., $R(s,a) = U(s)$ for any $a \in A$. With such a conversion, we can also obtain $f_{P^\pi} \succeq f_{V^\pi}$ in these cases.
- Our second category covers the exclusive case of the first category. For example, consider an environment, where two actions a_1, a_2 lead to the same new state s' from state s but gain different rewards. In such a case, the reward is independent of the dynamics caused by s, a. We consider that such a case is minority in the ones of interest.

□

The theorem declares how the three policy abstractions are related to each other in the sense of abstraction fineness with the two extreme cases (f_Θ, f_0) for reference. The coarser the abstraction is, the more the original policy space is abstracted. In Tab. 1, we summarize the properties of different abstractions, regarding abstraction criteria, fineness, and task relevance. The primary conclusion is that there is an inverse relation between abstraction fineness and task relevance. Except for the two extreme cases (f_Θ, f_0) that are totally task-independent, the policy abstraction becomes more task-relevant as the abstraction criterion concerns more policy features related to the learning task. f_π concerns the policy behavior in the learning task. f_π is coarser than f_Θ since the same policy behavior can be realized by non-unique policy parameters. Taking one step closer to the task, f_{P^π} cares about the state transition dynamics induced by policy behavior. f_{P^π} is coarser than f_π as different behaviors may induce the same transition distribution. f_{V^π} further involves the rewards of long-term dynamics, thus is the most task-relevant and coarsest among the three types of policy abstraction. In Tab 2, we summarize the original abstraction criteria used in various downstream tasks (Third Column) and provide more promising policy abstraction options for corresponding downstream tasks based on our policy abstraction theory (Fourth Column).

4 POLICY METRIC

The policy abstractions allow us to aggregate policies through an equivalence relation. Nevertheless, exact equivalence is rarely encountered in continuous policy space (e.g., the usual case with neural policies), thus useful abstraction can be challenging to obtain. Moreover, the equivalence relation offers only qualitative (i.e., binary) outcomes and is incapable of measuring the similarity between policies, which is significant to policy representation learning. To this end, we generalize the policy abstractions to policy metrics which quantitatively measure the distance between two policies.

4.1 Theoretical Formulation

Corresponding to the three types of policy abstraction, we define the following three policy metrics:

Definition 4. Given an MDP, a ground policy space Π, a state distribution $p(s)$ and a distribution (pseudo-)metric $D(\cdot,\cdot)$, for any two policies $\pi_i, \pi_j \in \Pi$, we define three policy metrics as follows:

1. A distribution-irrelevance metric:
   $d_\pi(\pi_i, \pi_j) = \mathbb{E}_{s\sim p(s)}\left[D\left(\pi_i(a \mid s), \pi_j(a \mid s)\right)\right]$.
2. An influence-irrelevance metric:
   $d_{P^\pi}(\pi_i, \pi_j) = \mathbb{E}_{s\sim p(s)}\left[D\left(P^{\pi_i}(s' \mid s), P^{\pi_j}(s' \mid s)\right)\right]$.
3. A value-irrelevance metric:
   $d_{V^\pi}(\pi_i, \pi_j) = \mathbb{E}_{s\sim p(s)}\left[D\left(Z^{\pi_i}(s), Z^{\pi_j}(s)\right)\right]$.

These metrics adhere to the same abstraction criteria as in Definition 2, i.e., the irrelevance regarding action distribution, influence, and value, measuring the similarity of policies by the distance at respective levels. Unlike the binary outcomes of equivalence relations (i.e., $d_\pi^{\text{Eq}}(\pi_i, \pi_j) = 0$ if $f_\pi(\pi_i) = f_\pi(\pi_j)$, and 1 otherwise), the metrics are continuous and offer more detailed information for policy comparison and low-dimensional policy representation learning following specific policy abstractions.

4.2 Estimating Policy Metrics via Maximum Mean Discrepancy

Given a tractable metric D and a state distribution p, the policy metrics (i.e., d_π, d_{P^π}, d_{V^π}) can be calculated exactly if the probability distributions (i.e., π, P^π, Z^π) are available. However, this is usually infeasible in practice; instead, in more regular cases, only

finite samples of policy interaction are available. Although the empirical distributions can be estimated in simple MDPs where the state-action space is finite (as in Appendix B), unfortunately, approximating these distributions and computing these metrics are non-trivial, especially with high-dimensional continuous state-action space.

Therefore, we estimate the policy metrics directly from the samples, bypassing estimating the empirical distributions (i.e., $\widetilde{\pi}$, $\widetilde{P}^{\pi}$, $\widetilde{Z}^{\pi}$). In particular, we adopt MMD [7, 18] as the distribution metric, i.e., let D be D_{MMD}. MMD measures the maximum value of the mean discrepancy of two distributions regarding all possible functions in a predefined family. Conventionally, let the class of functions $h : X \to \mathbb{R}$ be a unit ball in a Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ associated with a continuous kernel $k(\cdot,\cdot)$ on X. Let p, q be two distributions defined on X, and let x, x' and y, y' be i.i.d. samples from p and q respectively. The D_{MMD} is defined as:

$$\begin{aligned} D_{\text{MMD}}(p, q; \mathcal{H}) &= \sup_{h \in \mathcal{H}: \|h\|_{\mathcal{H}} \leq 1} \left(\mathbb{E}_{x\sim p}[h(x)] - \mathbb{E}_{y\sim q}[h(y)]\right) \\ &= \|\mu_p - \mu_q\|_{\mathcal{H}} \\ &= \left(\mathbb{E}_{x,x'}[k(x,x')] + \mathbb{E}_{y,y'}[k(y,y')] - 2\mathbb{E}_{x,y}[k(x,y)]\right)^{\frac{1}{2}}, \end{aligned} \tag{6}$$

where $\mu_p = \int_X k(x,\cdot)p(\mathrm{d}x)$ is the mean embedding of p into $\mathcal{H}$[27]. Thus, D_{MMD} can be empirically estimated with samples $\{x_i\}_{i=1}^N \sim p$ and $\{y_i\}_{i=1}^M \sim q$:

$$\begin{aligned} \widetilde{D}^2_{\text{MMD}}(\{x_i\}, \{y_i\}; k) &= \frac{1}{N^2}\sum_{i,j} k(x_i, x_j) + \frac{1}{M^2}\sum_{i,j} k(y_i, y_j) \\ &- \frac{2}{NM}\sum_{i,j} k(x_i, y_j). \end{aligned} \tag{7}$$

According to Eq. 7, we can estimate the policy metrics $d_\pi, d_{P^\pi}, d_{V^\pi}$ empirically from the samples $\{a_i\}, \{s'_i\}, \{G_i\}$ of different policies respectively, under the sampled states $\{s_i\}$ for the expectation $\mathbb{E}_{p(s)}$. However, it is often impractical to obtain multiple samples under the same state. Thus, we resort to estimating the surrogates, e.g., $\hat{d}_{P^\pi}(\pi_i, \pi_j) = D(P^{\pi_i}(s,s'), P^{\pi_j}(s,s'))$ for d_{P^π}, where the joint distributions rather than the state-conditioned distributions are measured. By default, we use the Gaussian Kernel for computation.

5 POLICY REPRESENTATION LEARNING APPROACH

The next question concerned in practice is: *how can we learn the representation of RL policies (usually modeled by NNs) in a general way?* Based on the policy metrics introduced above, we propose a policy representation learning approach by following the principle of Deep Metric Learning.

5.1 Learning Policy Representation by Embedding Alignment

The previously proposed policy metrics quantify the relationship between policies based on different policy abstraction criteria. To enable a unified policy representation learning function for different policy metrics, we propose the *alignment loss*. It minimizes the difference between the distances of two policies in the representation space and in the policy metric space. Specifically, the alignment loss formalizes with a policy representation function f_ψ.

$$\mathcal{L}_{\text{AL}}(\psi) = \mathbb{E}_{\pi,\pi'\in\Pi}\left[\left(\left\|f_\psi(\pi) - f_\psi(\pi')\right\|_2 - \eta d_*(\pi,\pi')\right)^2\right], \tag{8}$$

Algorithm 1 Proximal Policy Optimization with Policy Representations (the differences to classic PPO are highlighted in blue)

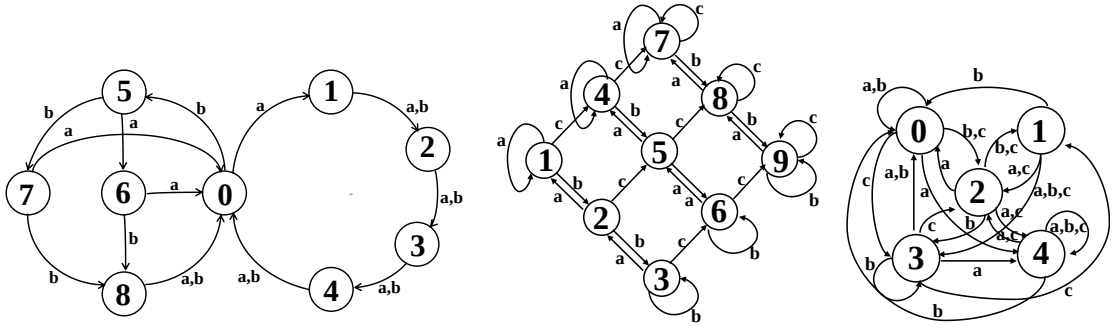
1: Initialize policy π_θ, PeVFA $\mathbb{V}_\beta$, policy representation function f_ψ and experience buffer $\mathbb{D}$
2: Set rollout trajectory number n and update interval U
3: **for** iteration $t = 0, 1, \ldots$ **do**
4: # Store policy and interaction experiences
5: Rollout policy π_{θ_t} in the environment and store policy and experiences $(\pi_{\theta_t}, \mathcal{D}_{\pi_t})$ in buffer $\mathbb{D}$ where trajectories $\mathcal{D}_{\pi_t} = \{\tau_i\}_{i=1}^n$
6: # PeVFA training policy representation training
7: **if** $t \,\%\, U = 0$ **then**
8: Update PeVFA parameter β with previous policy data $\{\mathcal{D}_{\pi_i}\}_{i=0}^{t-1}$
9: $\mathcal{L}_{\mathbb{V}}(\beta) = \sum_{\mathcal{D}_{\pi_i}\in\mathbb{D}} \mathbb{E}_{s,a,G_s\sim\mathcal{D}_{\pi_i}}\left[(G_s - \mathbb{V}_\beta(s, \chi_{\pi_i}))^2\right]$
10: Update policy representation function parameter ψ with policy data $\mathbb{D}$ (Eq.8)
11: **end if**
12: # PPO updates for π_t based on policy representation
13: Update PeVFA parameter β with current policy data $\mathcal{D}_{\pi_t}$ $\mathcal{L}_{\mathbb{V}}(\beta) = \mathbb{E}_{s,a,G_s\sim\mathcal{D}_{\pi_t}}\left[(G_s - \mathbb{V}_\beta(s, \chi_{\pi_t}))^2\right]$
14: Update policy parameter θ_t according to proximal policy improvement objective, and obtain the new policy $\pi_{\theta_{t+1}}$
15: **end for**

where we consider $d_* \in \{d_\pi, d_{P^\pi}, d_{V^\pi}\}$ and η is the weight for scaling. $\mathcal{L}_{\text{AL}}(\psi)$ consists of two metrics, i.e., the L_2 distance function ($\|\cdot\|_2$) and the policy metric ($d_*(\cdot,\cdot)$). Intuitively, minimizing the alignment loss is to align the two metrics by optimizing the policy representation function f_ψ. Thus, different policy representation functions which map $\pi \in \Pi$ to $\chi_\pi = f_\psi(\pi) \in \mathcal{X}$ preserve features corresponding to abstraction criteria reflected by the policy metric. With the empirical policy metrics provided in the previous section, the training of policy representation is straightforward with a differentiable function f_ψ by optimizing the alignment loss. Thus, a key problem is the implementation of the policy representation function f_ψ, which is detailed in the next subsection.

5.2 Realizing the Policy Representation Function

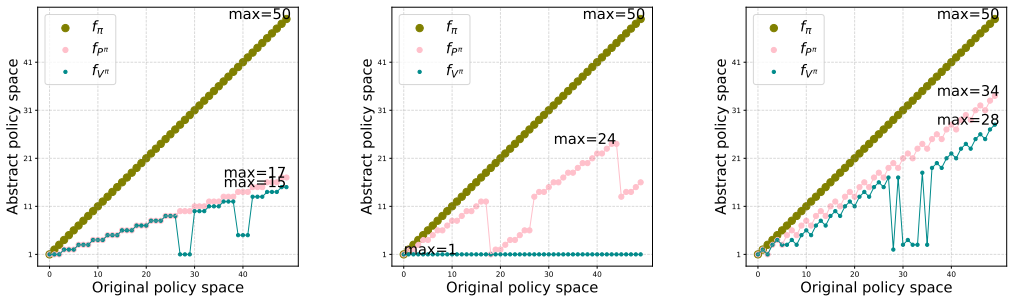
Intuitively, the realization of the policy representation function concerns two aspects: 1) the choice of policy data and 2) the construction of f_ψ (i.e., how policy data is encoded).

For the first aspect, we focus on parameterized policy π_θ (typically by a neural network) and use policy parameter θ as the policy data. One may recall that θ itself can be viewed as the finest representation obtained by policy abstraction f_Θ in Tab. 1. Such an original representation (i.e., θ) is high-dimensional and highly nonlinear, offering no help in the compression and generalization of policy space. In addition, we are aware that in some cases the policy parameters may be not available, and thus the interaction experiences generated by the policy can be alternative policy data, as used in [8, 28]. Our policy representation learning approach is compatible with such alternatives with the need of possible slight modifications. For the second aspect, inspired by the works [17, 28, 32] on characterizing the parameters of deep networks, we adopt Layer-wise Permutation-invariant Encoder (LPE) as the implementation choice of f_ψ. To be specific, for the parameter $\theta = \{W_i, b_i\}_{i=0}^k$ of policy π, i.e., the weights and biases of k-layer MLP, the weight $W_i \in \mathbb{R}^{l_i \times l_{i+1}}$



(a) Nchain (b) Upworld (c) Random

Figure 1: The illustration of three example MDPs. *(a)* Nchain; *(b)* Upworld; *(c)* Random.



(a) Nchain (b) Upworld (c) Random

Figure 2: The illustration of different policy abstractions in three example MDPs. *(a)* Nchain; *(b)* Upworld; *(c)* Random. For each MDP, the x-axis shows 50 policies in the ground policy space, and the y-axis shows the corresponding abstract policy in the abstract policy space for each policy in the ground policy space based on different policy abstractions.

and bias $b_i \in \mathbb{R}^{1\times l_{i+1}}$ (l_i is the unit number of the i-layer; l_0 and l_k are for the input and output layers) are concatenated ($\oplus$) and transposed, followed by an MLP ($f_{\psi,i}$) and a mean-reduce operation ($\texttt{MR}$), resulting in a layer embedding z_i. Thereafter, the policy embedding is obtained by concatenating the embedding of each layer. Formally,

$$z_i = \texttt{MR}\left(f_{\psi,i}([W_i \oplus b_i]^\top)\right) = \frac{1}{l_i+1}\sum_{j=1}^{l_i+1} f_{\psi,i}\left(([W_i \oplus b_i]^\top)_{j,\cdot}\right),$$
$$\chi_{\pi_\theta} = f_\psi(\theta) = \bigoplus_{i=0}^{k} z_i. \tag{9}$$

Each row of $[W_i \oplus b_i]^\top$, indexing by the subscript $j,\cdot$, describes a transformation of the i-layer into the next layer. All the rows are fed into $f_{\psi,i}$ separately and are then averaged into z_i. In a consequence, the policy embedding serves as the compact representation of the policy network by summarizing the transformations made by the each layer of it. Significantly, LPE provides structure-aware representation, i.e., both the intra-layer and inter-layer structures are explicitly considered. It alleviates the difficulty of learning representation from the policy network parameters. Other advanced encoder structures are beyond the scope of this work and we leave them as future work.

Until now, we can update the parameters of LPE $\psi = \{\psi_i\}_{i=0}^k$ by optimizing Eq. 8 with the policy samples from some given set of policies and the empirical policy metrics estimated accordingly.

Table 3: Comparison of policy space size based on different policy abstractions across MDPs. $\zeta(\Pi)$ and $\zeta(f_*)$ denote original policy space size and abstract policy space size corresponding to three types of policy abstraction.

Domain	Policy Space			
	$\zeta(\Pi)$	$\zeta(f_\pi)$	$\zeta(f_{P^\pi})$	$\zeta(f_{V^\pi})$
Nchain	**19683**	**19683**	**81** ↓	**63** ↓
Upworld	**19683**	**19683**	**8748** ↓	**161** ↓
Random	**243**	**243**	**108** ↓	**82** ↓

Depending on the specific choice of policy metric, the policy representation is learned to reflect the policy abstraction in Tab. 1, starting from f_Θ and going downwards to the corresponding level.

6 EXPERIMENTS

To demonstrate the effectiveness and practical application value of the policy abstraction theory and representation learning method proposed in this work, we perform three sets of comparative experiments, aiming to systematically answer the following key questions:

1. *Can the proposed policy abstractions effectively compress the policy space and satisfy the partial ordering relationship proven in this work?*

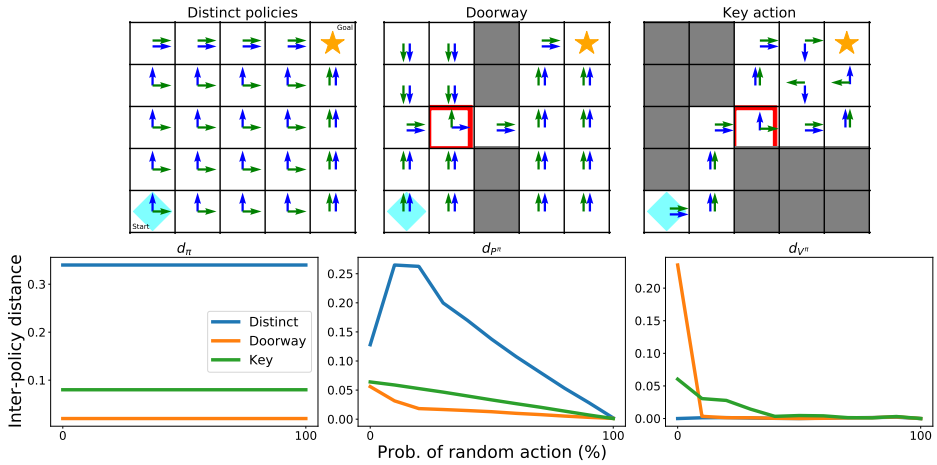


Figure 3: Policy comparison with different policy metrics in Gridworld. *Top Panel:* The illustration of three Gridworld MDPs and two deterministic policies (blue and green). *Bottom Panel:* The distance curves of the two policies measured by $d_\pi, d_{P^\pi}, d_{V^\pi}$ (y-axis), against the stochasticity of the environment (x-axis). d_{P^π} is able to distinguish the two policies across all the settings.

2. *Can the proposed policy metrics effectively measure the similarity between policies based on specific policy abstractions?*

3. *Can the proposed policy metric-based representation learning method effectively improve the performance of RL algorithms?*

Effectiveness Validation of Policy Abstraction Theory. To answer the first question, we apply three policy abstractions in three finite MDPs - *Nchain*, *Upworld*, and *Random* - to illustrate their properties [1, 3]. As shown in Fig. 1, each MDP consists of a set of states (denoted by numbers) and actions (denoted by letters), where the arrows between states indicate state transitions. Due to space limitations, we defer the detailed description of each MDP to Appendix C.1. Specifically, for each deterministic MDP, we can obtain its original policy space as well as the compressed policy space based on different policy abstractions.

From Tab. 3, it is clear that the distribution-irrelevance abstraction f_π perfectly captures the ground policy space of the non-parametric policies for the three MDPs, i.e., $\zeta(\Pi) = \zeta(f_\pi)$. However, the coarser abstractions, i.e., influence-irrelevance f_{P^π} and value-irrelevance f_{V^π}, aggregate policies based on the equivalence with respective concerns on the influence on the environment and value function induced by the policy. As a result, these abstractions yield a much smaller abstract policy space than the original policy space, i.e., $\zeta(f_{P^\pi}) < \zeta(\Pi)$, $\zeta(f_{V^\pi}) < \zeta(\Pi)$. From Tab. 3, we observe that $\zeta(f_{V^\pi}) < \zeta(f_{P^\pi}) < \zeta(f_\pi)$, which shows that the coarser the abstraction is, the more the original policy space is compressed. To further illustrate the partial ordering ($f_\pi \succeq f_{P^\pi} \succeq f_{V^\pi}$), we present the aggregation of the policy space for each MDP in Fig. 2. Specifically, considering that the original policy space is too extensive ($\zeta(\Pi) = 19683$ for Nchain and Upworld), we randomly select 50 policies for each MDP as the policy subspace $\hat{\Pi}$ to present the aggregation of policies. From Fig. 2, we observe that $\forall f_1, f_2 \in \{f_\pi, f_{P^\pi}, f_{V^\pi}\}$, if $f_1 \succeq f_2$, $\forall \pi_1, \pi_2 \in \hat{\Pi}$, $f_1(\pi_1) = f_1(\pi_2)$ implies $f_2(\pi_1) = f_2(\pi_2)$. More results can be found in Appendix C.1, and the above conclusions are applicable to the entire policy space. These results indicate that the proposed policy abstractions can achieve multi-level compression of the policy space while retaining critical policy features, thereby addressing the theoretical gap in current research on policy space compression.

Effectiveness Validation of Policy Metric. To answer the second question, we demonstrate how the distances of two policies measured by the corresponding policy metrics differ in several Gridworld MDPs. We use *Distinct Policies*, *Doorway* from [13] and design a new task, *Key Action* for prototypes of tasks with different features. Moreover, we increase the stochasticity of the environment for robustness evaluation. In particular, $\mathbb{E}_{s\sim p(s)} D(\cdot,\cdot)$ is calculated by averaging the absolute differences over all states. The illustrations and results are shown in Fig. 3, while more results can be found in Appendix C.2.

As shown in Fig. 3, the three policy metrics produce distinct results for any given pair of policies. Specifically, d_π can measure the difference in action distributions between policies, independent of their dynamics and outcomes (e.g., Doorway policies), and remain unaffected by increasing environmental stochasticity. Similarly, d_{P^π} and d_{V^π} focus on the differences in dynamics and outcomes between policies, respectively. Unlike d_π , as environmental stochasticity increases, d_{P^π} and d_{V^π} gradually degrade. These results demonstrate that the proposed policy metrics can effectively measure the similarity between policies based on specific policy abstractions, providing a solid foundation for policy comparison and policy representation learning.

Effectiveness Validation of Policy Representation Learning Method. To answer the third question, we propose to improve the classical PPO [25] algorithm through policy representations. Specifically, we draw on the core ideas of the PPO-PeVFA [28] and extend the traditional value function approximator ($V(s) \rightarrow \mathbb{V}(s, \chi_\pi)$) using policy representations χ_π, thereby improving the approximation and generalization of the value function approximator. Among them, policy representation is a core component of this framework, and its learning quality directly impacts algorithm performance. However, PPO-PeVFA [28] has obvious limitations in its policy representation learning mechanism: due to the lack of available policy abstraction theory, it primarily adopts End-to-End

Table 4: Max Average Return over 10 trials of 2M time steps (4M for Ant). The maximum value for each task is bolded. ± corresponds to half a std over trials. Conclusion: Our proposed algorithms perform the best on most tasks.

Environments	Original Algorithms	Verification Baselines			Ours		
	PPO	PPO-PeVFA (E2E)	PPO-PeVFA (CL)	PPO-PeVFA (RC)	PPO-PeVFA (d_π)	PPO-PeVFA (d_{P^π})	PPO-PeVFA (d_{V^π})
HalfCheetah	2621 ± 204.05	3171 ± 427.63	3725 ± 348.55	3175 ± 517.52	3341 ± 125.30	3750 ± 256.25	3935 ± 230.15
Hopper	1639 ± 247.15	2085 ± 310.91	2351 ± 231.11	2214 ± 360.78	2220 ± 220.35	2315 ± 151.80	2450 ± 152.31
Walker2d	1505 ± 261.88	1856 ± 305.51	2038 ± 315.51	2044 ± 316.32	2160 ± 162.35	2180 ± 166.45	2213 ± 186.54
Ant	2835 ± 116.67	3581 ± 185.43	4019 ± 162.47	3784 ± 268.99	3835 ± 265.35	3868 ± 286.36	3980 ± 112.22
InvDouPend	9344 ± 11.02	9357 ± 0.29	9355 ± 0.64	9355 ± 0.68	9358 ± 1.14	9358 ± 2.12	9420 ± 2.24
LunarLander	219 ± 5.33	238 ± 3.37	239 ± 3.70	234 ± 3.47	236 ± 2.32	239 ± 3.20	241 ± 2.54

(E2E) way and self-supervised learning principles (e.g., Contrastive Learning (CL) and Policy Recovery (RC)), which makes it difficult to ensure the relevance of the learned representation to the task goal. This work fills this research gap and provides an interpretable theoretical basis for policy representation learning. Here, considering that the core role of policy representation in the PPO-PeVFA [28] is to improve the generalization of the value function across policy spaces, we propose replacing the original self-supervised learning principles with the value-irrelevance abstraction principle. The algorithm pseudocode is detailed in Algorithm 1.

To demonstrate the effectiveness of our proposed method, we rigorously adhere to the experimental setup of PPO-PeVFA [28] and conduct systematic validation on OpenAI Gym [2] continuous control tasks. For all the baselines and variants used in this subsection, we utilize a normal-scale policy network with 2 layers and 64 units per layer. For PPO [25], the conventional value network $V(s)$ is a 2-layer 128-unit ReLU-activated MLP with state as input and value as output. For PPO-PeVFA [28], the PeVFA network $\mathbb{V}(s, \chi_\pi)$ takes as input state and policy representation, and the dimensionality of policy representation is set to 64. More experimental details and full learning curves can be found in Appendix C.3.

The experimental results in Tab. 4 show that PPO-PeVFA significantly outperforms its original counterpart (PPO[25]). This demonstrates the superiority and research value of the proposed policy representation method. Moreover, our proposed **PPO-PeVFA(d_{V^π})** exhibits superior performance compared to the **E2E**, **CL**, and **RC** baseline algorithms across most environments, demonstrating the effectiveness of our policy representation learning method. Additionally, d_{V^π} outperforms both d_π and d_{P^π}, which further confirms the significance of policy abstraction theory in guiding policy representation learning: value-irrelevance abstraction (f_{V^π}) can effectively preserve the policy's value information, leading to improved performance on value function approximation and generalization tasks. For a closer look, we report the aggregated results on all tasks except *InvDouPend* and *LunarLander* (since most algorithms obtain near-optimal performance on them) in Fig.4. Specifically, we observe that **RC** performs similarly to **PPO-PeVFA (d_π)**, mainly because both focus on policy distribution. **PPO-PeVFA(d_{P^π})** demonstrates intermediate performance, and **PPO-PeVFA(d_{V^π})** outperforms all comparative methods in both learning efficiency and final convergence performance. These results demonstrate the critical role of policy abstraction theory in policy representation learning.

In this work, we focus on the task of value function approximation and generalization, validating the practical value of policy abstraction theory and representation methods in this task. It is worth noting that, as mentioned in the introduction, policy representation has potential applications in tasks such as Opponent Modeling [8], Policy Adaptation [22, 24], Behavioral Characterization [13], etc. However, these extended studies are far beyond the scope of this work, and we will further explore the application potential of policy representation in various downstream tasks in future research.

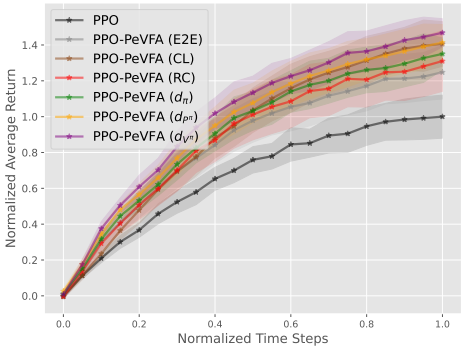


Figure 4: Aggregated averaged returns over four MuJoCo tasks. The returns are normalized based on the results of PPO in Tab.4.

7 CONCLUSION & LIMITATIONS

This study proposes a unified policy abstraction theory, systematically encompassing its definition, criteria, properties, and partial ordering relations. Further, we generalize policy abstractions to policy metrics and propose a unified optimization objective for policy representation, as well as an efficient implementation method tailored to various abstraction criteria. Systematic experimental verification demonstrates the great potential of policy abstraction in compressing policy space, characterizing policy differences, and conveying policy generalization, which is of great significance for the development and application of RL algorithms. Although this study has made significant progress in the field of policy abstraction, the proposed unified theory and method represent only an initial exploration in this field. In future work, we will further refine the theoretical system of policy abstraction and explore other promising representation learning principles.

ACKNOWLEDGMENTS

This work is supported by the MoE Key Laboratory of Brain-inspired Intelligent Perception and Cognition, University of Science and Technology of China (Grant Nos. 2521006). This work is supported by the National Natural Science Foundation of China (Grant Nos. 62422605, 62533021, 62541610, 92370132).

REFERENCES

[1] D. Abel, D. Hershkowitz, and M. Littman. 2016. Near optimal behavior via approximate state abstraction. In *ICML*. PMLR, 2915–2923.
[2] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. 2016. OpenAI Gym. *arXiv preprint arXiv:1606.01540* (2016).
[3] R. Dearden, N. Friedman, and S. Russell. 1998. Bayesian Q-learning. *Aaai/iaai* 1998 (1998), 761–768.
[4] F. Faccio, R. Aditya, H. Vincent, J. Harb, and J. Schmidhuber. 2022. General policy evaluation and improvement by learning to identify few but crucial states. *arXiv preprint arXiv:2207.01566* (2022).
[5] F. Faccio, L. Kirsch, and J. Schmidhuber. 2020. Parameter-based value functions. *arXiv preprint arXiv:2006.09226* (2020).
[6] F. Giunchiglia and T. Walsh. 1992. A theory of abstraction. *Artificial intelligence* 57, 2-3 (1992), 323–389.
[7] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. 2012. A Kernel Two-Sample Test. *JMLR* 13 (2012), 723–773.
[8] A. Grover, M. Al-Shedivat, J. Gupta, Y. Burda, and H. Edwards. 2018. Learning Policy Representations in Multiagent Systems. In *ICML*, Vol. 80. 1797–1806.
[9] N. Hansen and A. Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation* 9, 2 (2001), 159–195.
[10] J. Harb, T. Schaul, D. Precup, and P. Bacon. 2020. Policy Evaluation Networks. *arXiv preprint arXiv:2002.11833* (2020).
[11] J. Ho and S. Ermon. 2016. Generative adversarial imitation learning. *NeurIPS* 29 (2016), 4565–4573.
[12] H. Jeffreys. 1946. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences* 186, 1007 (1946), 453–461.
[13] A. Kanervisto, T. Kinnunen, and V. Hautamäki. 2020. General Characterization of Agents by States they Visit. *arXiv preprint arXiv:2012.01244* (2020).
[14] X. Ma, J. Gupta, and M. Kochenderfer. 2020. Normalizing Flow Model for Policy Representation in Continuous Action Multi-agent Systems. In *AAMAS*. 1916–1918.
[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.
[16] M. Mutti, S. Col, and M. Restelli. 2021. Reward-Free Policy Space Compression for Reinforcement Learning. In *Unsupervised Reinforcement Learning Workshop at ICML*.
[17] A. Navon, A. Shamsian, I. Achituve, E. Fetaya, G. Chechik, and H. Maron. 2023. Equivariant architectures for learning in deep weight spaces. In *International Conference on Machine Learning*. PMLR, 25790–25816.
[18] T. Nguyen-Tang, S. Gupta, and S. Venkatesh. 2021. Distributional Reinforcement Learning via Moment Matching. In *AAAI*. 9144–9152.
[19] A. Pacchiano, J. Parker-Holder, Y. Tang, K. Choromanski, A. Choromanska, and M. I. Jordan. 2020. Learning to Score Behaviors for Guided Policy Optimization. In *ICML*, Vol. 119. 7445–7454.
[20] B. Peng, X. Li, J. Gao, J. Liu, K. Wong, and S. Su. 2018. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176* (2018).
[21] M. Puterman. 2014. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.
[22] R. Raileanu, M. Goldstein, A. Szlam, and R. Fergus. 2020. Fast Adaptation to New Environments via Policy-Dynamics Value Functions. In *ICML*, Vol. 119. 7920–7931.
[23] H. Royden. 1968. Real analysis / H. L. Roiden.
[24] T. Sang, H. Tang, Yi Ma, J. Hao, Y. Zheng, Z. Meng, B. Li, and Z. Wang. 2022. PAnDR: Fast Adaptation to New Environments from Offline Experiences via Decoupling Policy and Environment Representations. *arXiv preprint arXiv:2204.02877* (2022).
[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[26] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine. 2019. AVID: Learning Multi-Stage Tasks via Pixel-Level Translation of Human Videos. *arXiv preprint arXiv:1912.04443* (2019).
[27] A. Smola, A. Gretton, L. Song, and B. Schölkopf. 2007. A Hilbert Space Embedding for Distributions. In *ALT (Lecture Notes in Computer Science, Vol. 4754)*. 13–31.
[28] H. Tang, Z. Meng, J. Hao, C. Chen, D. Graves, D. Li, C. Yu, H. Mao, W. Liu, Y. Yang, W. Tao, and L. Wang. 2020. What About Inputing Policy in Value Function: Policy Representation and Policy-extended Value Function Approximator. *arXiv preprint arXiv:2010.09536* (2020).
[29] E. Todorov, T. Erez, and Y. Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.
[30] J. Urain, D. Tateo, T. Ren, and J. Peters. 2020. Structured Policy Representation: Imposing Stability in arbitrarily conditioned dynamic systems. *arXiv preprint arXiv:2012.06224* (2020).
[31] Y. Zheng, Z. Meng, J. Hao, Z. Zhang, T. Yang, and C. Fan. 2018. A Deep Bayesian Policy Reuse Approach Against Non-Stationary Agents. In *NeurIPS 2018*. 962–972.
[32] A. Zhou, K. Yang, K. Burns, C. Adriano, Y. Jiang, S. Sokota, J. Kolter, and F.Chelsea. 2024. Permutation equivariant neural functionals. *Advances in Neural Information Processing Systems* 36 (2024).

A MORE CONTENT ON POLICY ABSTRACTION THEORY

A.1 Other Policy Abstractions

In this paper, we also propose three other policy abstractions in Definition 5. Before introducing the definitions of additional policy abstractions, we make some necessary notations. We use $d^{\pi,k}(\cdot)$ to denote the distribution of state when policy π performs k steps from initial states regarding the initial state distribution ρ_0. The discounted state visitation distribution from initial states regarding ρ_0 is defined as $d^{\pi}(s') = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d^{\pi,t}(s')$ for any $s' \in S$. Additionally, we use $d_s^{\pi,k}(\cdot)$ to denote the distribution of state when policy π performs k steps from any state s. The discounted state visitation distribution from any state s is defined as $d_s^{\pi}(s') = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t d_s^{\pi,t}(s')$ for any $s' \in S$.

Definition 5. Given an MDP $\langle S, A, P, R, \gamma\rangle$ and a ground policy space Π, for any two policies $\pi_i, \pi_j \in \Pi$, we define three more policy abstractions as follows:

1. An influence-irrelevance abstraction ($f_{d_s^{\pi}}$) is such that for all $s, s' \in S$, $f_{d_s^{\pi}}(\pi_i) = f_{d_s^{\pi}}(\pi_j)$ implies that $d_s^{\pi_i}(s') = d_s^{\pi_j}(s')$.
2. An influence-irrelevance abstraction ($f_{d^{\pi}}$) is such that for all $s \in S$, $f_{d^{\pi}}(\pi_i) = f_{d^{\pi}}(\pi_j)$ implies that $d^{\pi_i}(s) = d^{\pi_j}(s)$.
3. A value-irrelevance abstraction ($f_{J^{\pi}}$) is such that $f_{J^{\pi}}(\pi_i) = f_{J^{\pi}}(\pi_j)$ implies that $\mathbb{E}_{s_0\sim\rho_0}\left[V^{\pi_i}(s_0)\right] = \mathbb{E}_{s_0\sim\rho_0}\left[V^{\pi_j}(s_0)\right]$.

Similarly, we prove how the newly proposed policy abstractions are related to other abstractions we introduce in the main body of the paper.

Theorem 2 (Partial Ordering ($\succeq$))**.** Under the Definition 2, 3 and 5, if the reward function R depends only on state $s \in S$, we have ❶ $f_{\pi} \succeq f_{P^{\pi}} \succeq f_{d_s^{\pi}} \succeq f_{d^{\pi}} \succeq f_{J^{\pi}}$; ❷ $f_{\pi} \succeq f_{P^{\pi}} \succeq f_{d_s^{\pi}} \succeq f_{V^{\pi}} \succeq f_{J^{\pi}}$. An illustration is provided below:

$$\begin{array}{ccccccc} f_{\pi} & \succeq & f_{P^{\pi}} & \succeq & f_{d_s^{\pi}} & \succeq & f_{d^{\pi}} \\ & & & & \curlyvee & & \curlyvee \\ & & & & f_{V^{\pi}} & \succeq & f_{J^{\pi}} \end{array}$$

Proof. The partial ordering ($\succeq$) satisfies transitivity, thus, let us prove the Theorem 2 one by one in the following.

❶$f_{P^{\pi}} \succeq f_{d_s^{\pi}}$. Given an MDP M, two policies $\pi_i, \pi_j \in \Pi$. If $f_{P^{\pi}}(\pi_i) = f_{P^{\pi}}(\pi_j)$, with the Definition 2, we have $\forall s, s' \in S, P^{\pi_i}(s' \mid s) = P^{\pi_j}(s' \mid s)$. Since the initial state distribution are the same, and π_i, π_j follow two identical Markov chains, we derive $f_{P^{\pi}} \succeq f_{d_s^{\pi}}$.

❷$f_{d_s^{\pi}} \succeq f_{d^{\pi}}$. Given an MDP, two policies $\pi_i, \pi_j \in \Pi$. If $f_{d_s^{\pi}}(\pi_i) = f_{d_s^{\pi}}(\pi_j)$, with the Definition 5, we have $\forall s, s' \in S$, $d_s^{\pi_i}(s') = d_s^{\pi_j}(s')$. Furthermore, we derive,

$$\forall s_0 \in \rho_0, s' \in S,\ d_{s_0}^{\pi_i}(s') = d_{s_0}^{\pi_j}(s'). \tag{10}$$

Thus, $f_{d_s^{\pi}} \succeq f_{d^{\pi}}$.

❸$f_{d^{\pi}} \succeq f_{J^{\pi}}$. Given an MDP, two policies $\pi_i, \pi_j \in \Pi$. If $f_{d^{\pi}}(\pi_i) = f_{d^{\pi}}(\pi_j)$, with the Definition 5, we have $\forall s \in S$, $d^{\pi_i}(s) = d^{\pi_j}(s)$. When the reward function R depends only on state $s \in S$, we have,

$$\begin{aligned} J(\pi_i) &= (1-\gamma)^{-1}\mathbb{E}_{s\in d^{\pi_i}}[R(s)], \\ J(\pi_j) &= (1-\gamma)^{-1}\mathbb{E}_{s\in d^{\pi_j}}[R(s)]. \end{aligned} \tag{11}$$

Thus, $f_{d^{\pi}} \succeq f_{J^{\pi}}$.

❹$f_{d_s^{\pi}} \succeq f_{V^{\pi}}$. Given an MDP, two policies $\pi_i, \pi_j \in \Pi$. If $f_{d_s^{\pi}}(\pi_i) = f_{d_s^{\pi}}(\pi_j)$, with the Definition 5, we have $\forall s, s' \in S$, $d_s^{\pi_i}(s') = d_s^{\pi_j}(s')$. When the reward function R depends only on state $s' \in S$, we have,

$$\begin{aligned} \forall s \in S,\ V^{\pi_i}(s) &= (1-\gamma)^{-1}\mathbb{E}_{s'\in d_s^{\pi_i}}[R(s')], \\ V^{\pi_j}(s) &= (1-\gamma)^{-1}\mathbb{E}_{s'\in d_s^{\pi_j}}[R(s')]. \end{aligned} \tag{12}$$

Thus, $f_{d_s^{\pi}} \succeq f_{V^{\pi}}$.

❺$f_{V^{\pi}} \succeq f_{J^{\pi}}$. Given an MDP, two policies $\pi_i, \pi_j \in \Pi$. If $f_{V^{\pi}}(\pi_i) = f_{V^{\pi}}(\pi_j)$, with the Definition 2, we have $\forall s \in S, V^{\pi_i}(s) = V^{\pi_j}(s)$. Furthermore, we derive,

$$\begin{aligned} &\forall s_0 \in \rho_0,\ V^{\pi_i}(s_0) = V^{\pi_j}(s_0), \\ &\mathbb{E}_{s_0\sim\rho_0}\left[V^{\pi_i}(s_0)\right] = \mathbb{E}_{s_0\sim\rho_0}\left[V^{\pi_j}(s_0)\right]. \end{aligned} \tag{13}$$

Thus, $f_{V^{\pi}} \succeq f_{J^{\pi}}$.

In particular, when the reward function R depends on both state $s \in S$ and action $a \in A$, π_i may not be equivalent to π_j. The $f_{d^{\pi}} \succeq f_{J^{\pi}}$ and $f_{d_s^{\pi}} \succeq f_{V^{\pi}}$ are not hold. Connecting Definition 2,5 and Theorem 2, we summarize the properties of different policy abstractions in Tab. 1. □

B MORE CONTENT ON POLICY METRIC

Definition 6 (Metrics [23])**.** Let X be a non-empty set of data elements and a **metric** is a real-valued function $d\colon X \times X \to [0, \infty)$ such that for all $x, y, z \in X$:

(1) $d(x, y) = 0 \iff x = y$;
(2) $d(x, y) = d(y, x)$;
(3) $d(x, y) \le d(x, z) + d(z, y)$.

A **pseudo-metric** d is a metric with the first condition replaced by $x = y \implies d(x, y) = 0$. The combination $\langle X, d\rangle$ is called a metric space.

B.1 Estimating Policy Metrics via Jeffrey Divergence

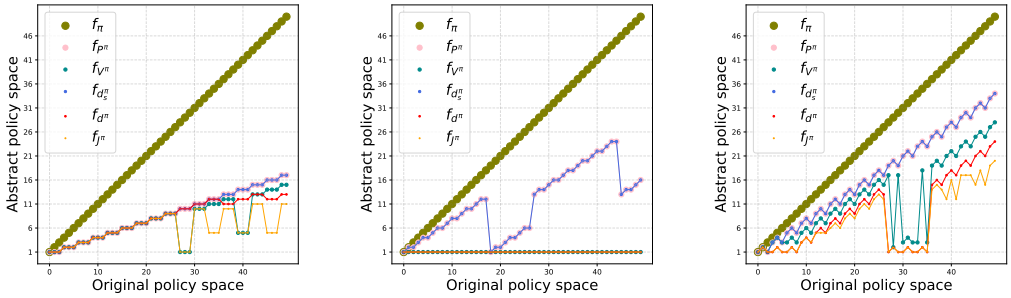
In simple MDPs where the state-action space is finite, we calculate the frequency distribution (i.e., $\widetilde{\pi}$, $\widetilde{P^{\pi}}$, $\widetilde{Z^{\pi}}$) using sufficient samples as an estimate of the exact probability distribution (i.e., π, P^{π}, Z^{π}). Then we use the Jeffreys Divergence [12] between empirical distributions as policy metrics (i.e., $d_{\pi}(\cdot,\cdot)$, $d_{P^{\pi}}(\cdot,\cdot)$, and $d_{V^{\pi}}(\cdot,\cdot)$).

$$\begin{aligned} d_{\pi}(\pi_i, \pi_j) &\approx D_{KL}\left(\widetilde{\pi}_i \| \widetilde{\pi}_j\right) + D_{KL}\left(\widetilde{\pi}_j \| \widetilde{\pi}_i\right), \\ d_{P^{\pi}}(\pi_i, \pi_j) &\approx D_{KL}\left(\widetilde{P}^{\pi_i} \| \widetilde{P}^{\pi_j}\right) + D_{KL}\left(\widetilde{P}^{\pi_j} \| \widetilde{P}^{\pi_i}\right), \\ d_{V^{\pi}}(\pi_i, \pi_j) &\approx D_{KL}\left(\widetilde{Z}^{\pi_i} \| \widetilde{Z}^{\pi_j}\right) + D_{KL}\left(\widetilde{Z}^{\pi_j} \| \widetilde{Z}^{\pi_i}\right). \end{aligned} \tag{14}$$

C MORE CONTENT ON EXPERIMENTS

C.1 Experimental Details of Policy Abstraction Theory

Experimental Setting. The Nchain is implemented with two loops that cover nine states and involve two actions, a and b. The agent receives a reward of 1 and 2 when transitioning to state 0 from state 4 and 8, respectively, and the reward for the remaining transitions is 0. With a slip probability of 0.2, the agent performs the opposite action. The Upworld is a 3×3 grid, which consists of nine states



(a) Nchain (b) Upworld (c) Random

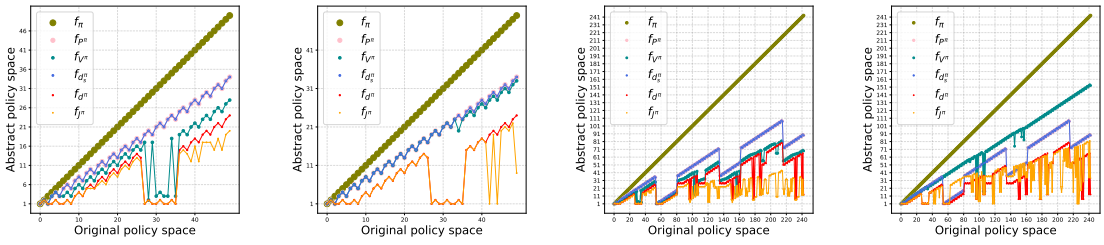
Figure 5: The illustration of different policy abstractions in three example domains. *(a)* Nchain; *(b)* Upworld; *(c)* Random. For each domain, the x-axis shows the top 50 policies in the ground policy space, and the y-axis shows the corresponding abstract policy in the abstract policy space for each policy in the ground policy space based on different policy abstractions.

Table 5: Comparison of policy space size based on different policy abstractions across domains (Nchain, Upworld, Random). $\zeta(\Pi)$ and $\zeta(f_*)$ denote the original policy space size and abstract policy space size corresponding to six policy abstractions.

Domain	Policy Space						
	$\zeta(\Pi)$	$\zeta(f_\pi)$	$\zeta(f_{P^\pi})$	$\zeta(f_{d_s{}^\pi})$	$\zeta(f_{d^\pi})$	$\zeta(f_{V^\pi})$	$\zeta(f_{J^\pi})$
Nchain	**19683**	**19683**	**81**	**81**	**69**	**63**	**33**
Upworld	**19683**	**19683**	**8748**	**8748**	**61**	**161**	**6**
Random	**243**	**243**	**108**	**108**	**80**	**82**	**44**

Table 6: Comparison of policy space size based on different policy abstractions for Random domain. $\zeta(\Pi)$ and $\zeta(f_*)$ denote the original policy space size and abstract policy space size corresponding to six policy abstractions.

Reward	Policy Space						
	$\zeta(\Pi)$	$\zeta(f_\pi)$	$\zeta(f_{P^\pi})$	$\zeta(f_{d_s{}^\pi})$	$\zeta(f_{d^\pi})$	$\zeta(f_{V^\pi})$	$\zeta(f_{J^\pi})$
R(s)	**243**	**243**	**108**	**108**	**80**	**82**	**44**
R(s,a)	**243**	**243**	**108**	**108**	**80**	**154**	**82**



(a) (b) (c) (d)

Figure 6: The illustration of different policy abstractions in the Random domain. *(a)* Random(R(s)/$\hat{\Pi}$); *(b)* Random(R(s,a)/$\hat{\Pi}$); *(c)* Random(R(s)/Π); *(d)* Random(R(s,a)/Π). The x-axis shows the ground policy space and the y-axis shows the corresponding abstract policy space based on different policy abstractions.

and three actions a, b, and c. The agent obtains a reward of 10 for transitions between states at the top of the grid, and the reward for the remaining transitions is 0. We implement the Random MDP with 5 states and 3 actions. In each state, each action transitions to one of two randomly selected states with a probability of 0.5. The reward function $R(s_t)$ is randomly initialized. In this study, we obtain the ground policy space of the three domains using the stochastic policies for Nchain and the deterministic policies for Upworld and Random MDP. Specifically, each policy in Nchain selects each action with a probability of 0, 1, or 0.5.

Experimental Results. Tab. 5 and Fig. 5 compare policy abstraction in two aspects: the size of the abstract policy space and the results of policy aggregation. Clearly, the experimental results indicate the

partial ordering (❶ $f_\pi \succeq f_{P^\pi} \succeq f_{d_s^\pi} \succeq f_{d^\pi} \succeq f_{J^\pi}$; ❷ $f_\pi \succeq f_{P^\pi} \succeq f_{d_s^\pi} \succeq f_{V^\pi} \succeq f_{J^\pi}$) between policy abstractions. The coarser the abstraction is, the more the original policy space is abstracted. In addition, to compare the two cases regarding the reward function, we consider two cases for the reward function in Random MDP, i.e., $R(s_t)$ and $R(s_t, a_t)$, which are randomly initialized. Tab. 6 and Fig. 6 report the experimental results of the two cases. Obviously, if the reward is independent of the dynamics caused by s, a, the $f_{d^\pi} \succeq f_{J^\pi}$ and $f_{d_s^\pi} \succeq f_{V^\pi}$ will not be held. As mentioned earlier, such a case is a minority in the ones of interest and has little impact on selecting an appropriate policy abstraction for a specific downstream problem. In Fig. **??**, we present the abstracted results of ground full policy space based on different levels of abstraction. In particular, we omit the distribution-irrelevance abstraction since $\zeta(\Pi) = \zeta(f_\pi)$. The abstraction results of the full policy space are consistent with the partial ordering of policy abstractions.

C.2 Experimental Details of Policy Metric

Experimental Setting. To quantitatively compare these policy metrics, we demonstrate how the distances of two policies (blue and green in Fig. 7) measured by the corresponding policy metrics differ in several Gridworld MDPs. We borrow *Distinct Policies*, *Doorway* from [13] and design a new environment, *Key Action* for simple prototypes of environments with different features. All three environments are *5×5* Gridworld, with the starting state and goal in the lower left and upper right grid, respectively. The discrete action space is {*up, down, left, right*}. We obtain the estimated value $v(\cdot)$ of each state by estimating 1 - `[expected number of step to goal when starting from a given state]`. About the *Key Action*, the agent obtain a positive reward only if it chooses *right* at the key state (i.e., marked by the red box in Fig. 7). Every agent rollouts 10k episodes for comparing different policy metrics. To be specific, a policy is represented by tensor $\pi \in \mathbb{R}^{5\times5\times4}$ and the distribution-irrelevance metric is defined as $d_\pi(\pi, \hat{\pi}) = \mathbb{E}_{\pi_{i,j,k}\in\pi, \hat{\pi}_{i,j,k}\in\hat{\pi}}\left[\left|\pi_{i,j,k} - \hat{\pi}_{i,j,k}\right|\right]$. Similarly, the state transition dynamics induced by policy is represented by tensor $P^\pi \in \mathbb{R}^{5\times5\times5}$ and the influence-irrelevance metric is defined as $d_{P^\pi}(\pi, \hat{\pi}) = \mathbb{E}_{P^\pi_{i,j,k}\in P^\pi, P^{\hat{\pi}}_{i,j,k}\in P^{\hat{\pi}}}\left[\left|P^\pi_{i,j,k} - P^{\hat{\pi}}_{i,j,k}\right|\right]$. Different from the d_{P^π} sampling-based estimation, we leverage the dynamic programming to learn the value function $V^\pi \in \mathbb{R}^{5\times5}$. Thus, the value-irrelevance metric is defined as $d_{V^\pi}(\pi, \hat{\pi}) = \mathbb{E}_{V^\pi_{i,j}\in V^\pi, V^{\hat{\pi}}_{i,j}\in V^{\hat{\pi}}}\left[\left|V^\pi_{i,j} - V^{\hat{\pi}}_{i,j}\right|\right]$. In addition, we also compare the two other policy abstractions proposed in Appendix A.1. Among them, we estimate the discounted state visitation distribution $d^\pi(s)$ via dividing visits to a state s by a total number of interactions (i.e., $d_{d^\pi}(\pi, \hat{\pi}) = \mathbb{E}_{d^\pi_{i,j}\in d^\pi, d^{\hat{\pi}}_{i,j}\in d^{\hat{\pi}}}\left[\left|d^\pi_{i,j} - d^{\hat{\pi}}_{i,j}\right|\right]$). Simply and naturally, based on the definition of f_{J^π} in Definition 5, we have $d_{J^\pi}(\pi, \hat{\pi}) = \left|\mathbb{E}_{s_0\sim\rho_0}\left[V^\pi(s_0)\right] - \mathbb{E}_{s_0\sim\rho_0}\left[V^{\hat{\pi}}(s_0)\right]\right|$.

Experimental Results. Fig. 7 shows the illustrations and the results of the five policy metrics. The d_{d^π} and d_{J^π} are newly added results compared to the original paper. We observe that the d_{d^π} cannot indicate the difference in dynamics between the two policies in *Key Action*. Like the d_{V^π}, the d_{J^π} measures the difference in the outcomes of the two policies but shows the poor robustness as the increase of stochasticity.

Table 7: Training details for our method. Grid search determines the best hyperparameter configuration for terms with multiple alternatives (i.e., {}).

Hyperparameters	Value
Actor Epoch	10
Critic Epoch	10
Policy Learning Rate	10^{-4}
Value Learning Rate	10^{-3}
Optimizer	Adam
Clipping Range Parameter (ϵ)	0.2
GAE Parameter (λ)	0.95
Discount Factor (γ)	0.99
On-policy Samples Per Iteration (n)	5 episodes or 2000 time steps
Update Interval (U)	Every 10 time steps
Batch Size	{64, 128}
Experience Buffer Size ($\mathbb{D}$)	200k (steps)
Policy Num Per Batch	{16, 32}
AL Learning Rate	10^{-3}
Gaussian Kernel (kernel_mul)	2.0
Gaussian Kernel (kernel_num)	5
AL Sample Pair Num (N, M)	1000
AL Scaling Weight η	{0.5, 1, 2}

C.3 Experimental Details of Policy Representation Method

Experimental Setting. Our experiments in this subsection are conducted in commonly adopted OpenAI Gym[1] continuous control tasks, i.e., *LunarLander* from Box2D and *HalfCheetah*, *Hopper*, *Walker2d*, *Ant* and *InvertedDoublePendulum* from MuJoCo [29]. We use the OpenAI Gym with version 0.9.1, the mujoco-py with version 0.5.4 and the MuJoCo products with version MJPRO131. Our codes are implemented with Python 3.6 and Tensorflow. We use Proximal Policy Optimization (PPO)[25] with Generalized Advantage Estimator (GAE) as our baseline lgorithm. For a fair comparison and clear evaluation, we perform no code-level optimization in our experiments, e.g., state standardization, reward scaling, gradient clipping, parameter sharing, etc. The algorithm PPO-PeVFA[28] is implemented based on PPO[25], which only differs in the replacement of the conventional value function network with the PeVFA network.

[1] http://gym.openai.com/

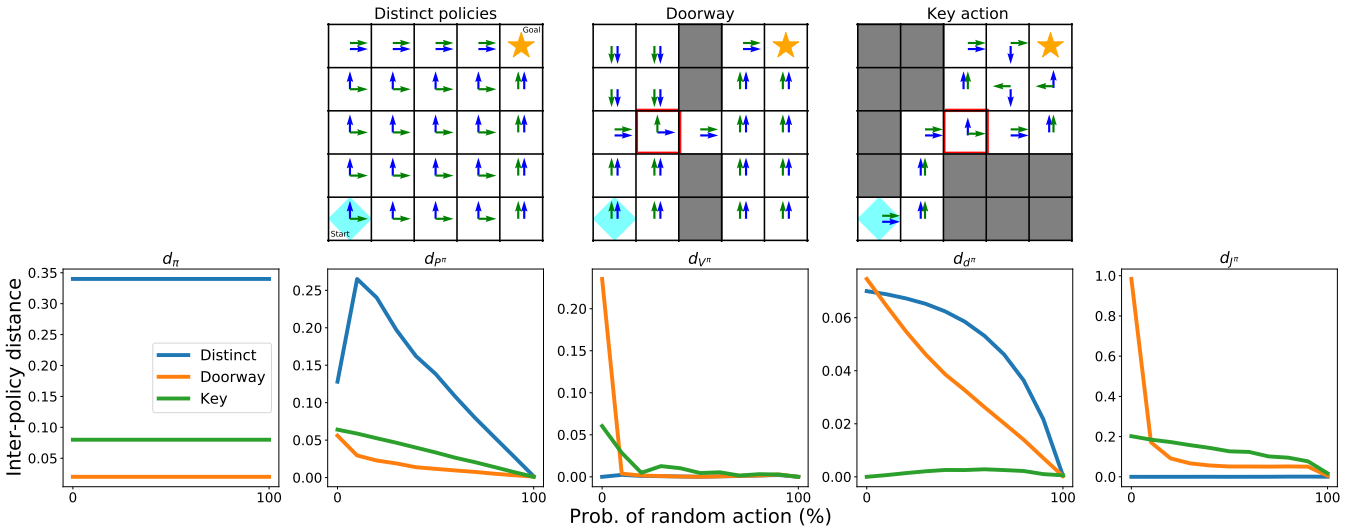


Figure 7: Policy comparison with different policy metrics in Gridworld. *Top Panel:* The illustration of three Gridworld MDPs and two deterministic policies (blue and green). *Bottom Panel:* The distance curves of the two policies measured by $d_\pi, d_{P^\pi}, d_{V^\pi}, d_{d^\pi}, d_{J^\pi}$ (y-axi), against the stochasticity of the environment (x-axi).

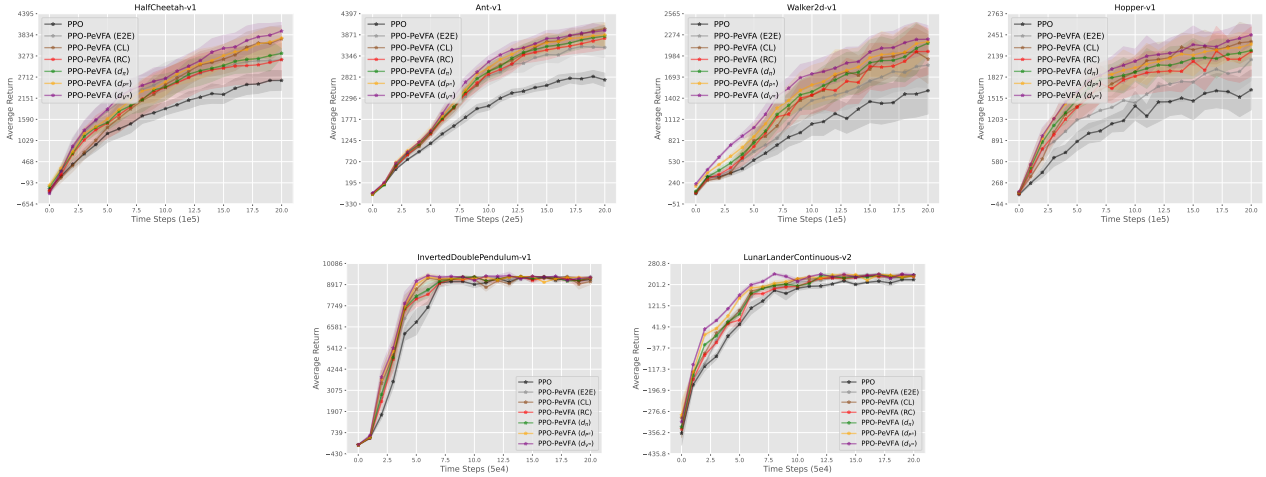


Figure 8: Learning curves for the OpenAI gym continuous control tasks. The shaded region represents half a standard deviation of the average evaluation over 10 trials. Curves are smoothed uniformly for visual clarity.