WHAT ABOUT TAKING POLICY AS INPUT OF VALUE FUNCTION: POLICY-EXTENDED VALUE FUNCTION APPROXIMATOR

Anonymous authors
Paper under double-blind review

ABSTRACT

The value function lies in the heart of Reinforcement Learning (RL), which defines the long-term evaluation of a policy in a given state. In this paper, we propose Policy-extended Value Function Approximator (PeVFA) which extends the conventional value to be not only a function of state but also an explicit policy representation. Such an extension enables PeVFA to preserve values of multiple policies in contrast to a conventional one with limited capacity for only one policy, inducing the new characteristic of *value generalization among policies*. From both the theoretical and empirical lens, we study value generalization along the policy improvement path (called local generalization), from which we derive a new form of Generalized Policy Iteration with PeVFA to improve the conventional learning process. Besides, we propose a framework to learn the representation of an RL policy, studying several different approaches to learn an effective policy representation from policy network parameters and state-action pairs through contrastive learning and action prediction. In our experiments, Proximal Policy Optimization (PPO) with PeVFA significantly outperforms its vanilla counterpart in MuJoCo continuous control tasks, demonstrating the effectiveness of value generalization offered by PeVFA and policy representation learning.

1 INTRODUCTION

Reinforcement learning (RL) has been widely considered as a promising way to learn optimal policies in many decision making problems (Mnih et al., 2015; Lillicrap et al., 2015; Silver et al., 2016; You et al., 2018; Schreck et al., 2019; Vinyals et al., 2019; Hafner et al., 2020). Lying in the heart of RL is the value function which defines the long-term evaluation of a policy. With function approximation (e.g., deep neural networks), a value function approximator (VFA) is able to approximate the values of a policy under large and continuous state spaces. As commonly recognized, most RL algorithms can be described as Generalized Policy Iteration (GPI) (Sutton & Barto, 1998). As illustrated in the left of Figure 1, at each iteration the VFA is trained to approximate the true values of current policy, regarding which the policy are improved. However, value approximation can never be perfect and its quality influences the effectiveness of policy improvement, thus raising a requirement for better value approximation (v. Hasselt, 2010; Bellemare et al., 2017; Fujimoto et al., 2018).

Since a conventional VFA only approximates the values (i.e., knowledge (Sutton et al., 2011)) for one policy, the knowledge learned from previously encountered policies is not preserved and utilized for future learning in an explicit way. For example in GPI, a conventional VFA cannot track the values of the changing policy by itself and has no idea of the direction of value generalization when approximating the values of a new policy. In this paper, we propose Policy-extended Value Function Approximator (PeVFA), which additionally takes an explicit policy representation as input in contrast to conventional VFA. PeVFA is able to preserve values for multiple policies and induces an appealing characteristic, i.e., value generalization among policies. We study the formal generalization and contraction conditions on the value approximation error of PeVFA, focusing specifically on value generalization along the policy improvement path which we call local generalization. Based on both theoretical and empirical evidences, we propose a new form of GPI with PeVFA (the right of Figure 1) which can benefit from the closer approximation distance induced by local value generalization under some conditions; thus, GPI with PeVFA is expected to be more efficient in consecutive value approximation along the policy improvement path.

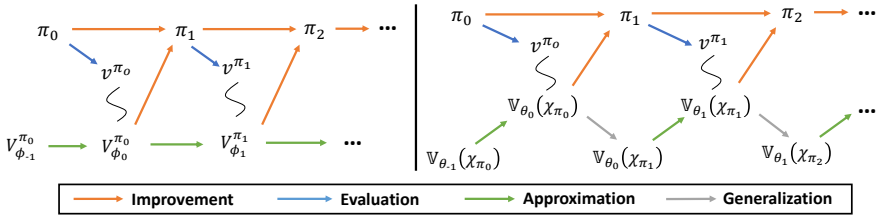


Figure 1: Generalized Policy Iteration (GPI) with function approximation. *Left*: GPI with conventional value function approximator V_ϕ^π. *Right*: GPI with PeVFA $\mathbb{V}_\theta(\chi_\pi)$ (Sec. 3) where extra generalization steps exist. The subscripts of policy π and value function parameters ϕ, θ denote the iteration number. The squiggle lines represent non-perfect approximation of true values.

Moreover, we propose a framework to learn effective policy representation for an RL policy from policy network parameters and state-action pairs alternatively, through contrastive learning and an auxiliary loss of action prediction. Finally, based on Proximal Policy Optimization (PPO), we derive a practical RL algorithm PPO-PeVFA from the above methods. Our experimental results demonstrate the effectiveness of both value generalization offered by PeVFA and policy representation learning. Our main contributions are summarized as follows:

- We propose PeVFA which improves generalization of values among policies and provide a theoretical analysis of generalization especially in local generalization scenario.
- We propose a new form of GPI with PeVFA resulting in closer value approximation along the policy improvement path demonstrated through experiments.
- To our knowledge, we are the first to learn a representation (low-dimensional embedding) for an RL policy from its network parameters (i.e., weights and biases).

2 Background

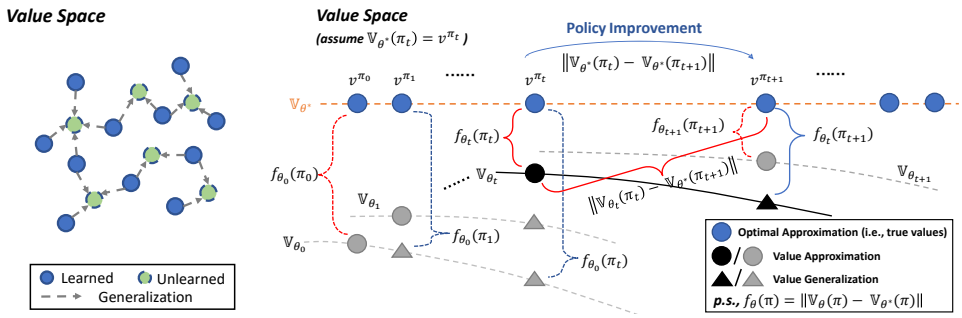
2.1 Reinforcement Learning

We consider a Markov Decision Process (MDP) defined as $\langle \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \gamma \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, r is the reward function, $\mathcal{P}$ is the transition function and $\gamma \in [0, 1)$ is the discount factor. The goal of an RL agent is to learn a policy $\pi \in \Pi$ where $\pi(a|s)$ is a distribution of action given state that maximizes the expected long-term discounted return. The *state-value function* $v^\pi(s)$ is defined in terms of the expected discounted return obtained through following the policy π from a state s: $v^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right]$ for all $s \in \mathcal{S}$ where $r_{t+1} = r(s_t, a_t)$. We use $V^\pi = v^\pi(\cdot)$ to denote the vector of values for all possible states. Value function is determined by policy π and environment models (i.e., $\mathcal{P}$ and r). For a conventional value function, policy is modeled implicitly within a table or a function approximator, i.e., a mapping from only state to value. One can refer to Appendix E.1 to see a more detailed description.

2.2 Extensions of Conventional Value Function

Schaul et al. (2015) introduced Universal Value Function Approximators (UVFA) that generalize values over goals in goal-conditioned RL. Similar ideas are also adopted for low-level learning of Hierarchical RL (Nachum et al., 2018). Such extensions are also studied in more challenging RL problems, e.g., opponent modeling (He & Boyd-Graber, 2016; Grover et al., 2018; Tacchetti et al., 2019), and context-based Meta-RL (Rakelly et al., 2019; Lee et al., 2020). General value function (GVF) in (Sutton et al., 2011) are proposed as a form of general knowledge representation through cumulants instead of rewards. In an unified view, each approach generalizes different aspects of the conventional VFA, focusing on different components of the vector form *Bellman equation* (Sutton & Barto, 1998) expanded on as discussed in Appendix E.2.

Concurrent to our work, Raileanu et al. (2020) proposes Policy-Dynamics Value Function (PDVF) for value generalization among policy and task to adapt quickly to new tasks. Their work can be viewed as a specific case of global generalization discussed further in Sec 3, and we focus on local generalization to improve learning efficiency in single tasks.



(a) Global generalization (b) Local generalization

Figure 2: Illustrations of value generalization with PeVFA among policies. Each circle represents the value function of a policy. (a) *Global generalization*: values can generalize to unlearned policies ($\pi' \in \Pi_1$) from already learned policies ($\pi \in \Pi_0$). (b) *Local generalization*: values of previous policies ($\{\pi_i\}_{i=0}^t$) can generalize to the successive policy (π_{t+1}) along policy improvement path.

3 Policy-extend Value Function Approximator

In this paper, we propose Policy-extended Value Function Approximator (PeVFA), an extension of the conventional value function that explicitly takes policy (representation) as input. Formally, consider a function $g : \Pi \to \mathcal{X} = \mathbb{R}^n$ in a general form that maps any policy π to a n-dimensional representation $\chi_\pi = g(\pi) \in \mathcal{X}$. The policy-extended value function $\mathbb{V} : \mathcal{S} \times \mathcal{X} \to \mathbb{R}$, defines the values over state and policy space:

$$\mathbb{V}(s, \chi_\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s \right], \text{ for all } s \in \mathcal{S}, \pi \in \Pi. \tag{1}$$

When only one policy π is considered, $\mathbb{V}(s, \chi_\pi)$ is equivalent in definition to a conventional value function $v^\pi(s)$. Similarly, we can define policy-extended action-value function $\mathbb{Q}(s, a, \chi_\pi)$. We use $\mathbb{V}(s, \chi_\pi)$ for demonstration in this paper.

The key point is, $\mathbb{V}(s, \cdot)$ is not defined for a specific policy and is able to preserve the values of multiple policies. With function approximation, a PeVFA is expected to approximate values among policy space, i.e., $\{v^\pi(s)\}_{\pi \in \Pi}$. Formally, with a limited parameter space Θ, we consider the optimal parameter θ^* with minimum approximation error over all possible states and policies as:

$$\theta^* = \arg\min_{\theta \in \Theta} F(\theta, \Pi), \text{ where } F(\theta, \Pi) = \max_{\pi \in \Pi} \|\mathbb{V}_\theta(\pi) - V^\pi\| = \max_{\pi \in \Pi} \|\mathbb{V}_\theta(\cdot, \pi) - v^\pi(\cdot)\|_\infty,$$

where $F(\theta, \Pi)$ is the overall approximation error of $\mathbb{V}_\theta$ which is defined as the maximum L_∞ norm of value vector distance among Π. Therefore, the learning process of a PeVFA $\mathbb{V}_\theta$ is $\theta \to \theta^*$.

As commonly recognized, a conventional value function approximator (VFA) can generalize the values to unseen states or state-action pairs after properly training on known states and state-action pairs under a specific policy. Beyond this, PeVFA provides an appealing characteristic that allows values to be generalized among policies, evaluating new policies with the knowledge of old ones. In the following, we study the characteristic of value generalization theoretically (Sec. 3.1) and provide some empirical evidences (Sec. 3.2), from which we finally introduce a new form of GPI (Sec. 3.3).

3.1 Value Generalization among Policies

Concretely, we introduce two scenarios of value generalization: global generalization and local generalization, which are closely related to many RL problems. An illustration is show in Figure 2. In this section, we first focus on a general two-policy case and study whether the learned value approximation of one policy can generalize to that of the other one. We propose a common form of generalization approximation error (Theorem 1) and analyze the condition of generalization contraction (Corollary 1). Further, we go to the two generalization scenarios as illustrated in Figure 2 and especially introduce the closer approximation distance along the policy improvement path induced by PeVFA (Corollary 2), from which a more efficient form of GPI can be derived (Sec. 3.3).

Formally, consider two policies π_1 and π_2 and a PeVFA $\mathbb{V}_\theta$ to approximate the values of them. For convenience of demonstration, we consider an identical representation space (i.e., $\mathcal{X} = \Pi$) and an unlimited parameter space Θ where zero approximation error can be achieved, i.e., $F(\theta^*, \Pi) = 0$. We then define the *approximation loss* f of $\mathbb{V}_\theta$ for each policy $\pi \in \Pi$ as $f_\theta(\pi) = \|\mathbb{V}_\theta(\pi) - \mathbb{V}_{\theta^*}(\pi)\| = \|\mathbb{V}_\theta(\cdot, \pi) - \mathbb{V}_{\theta^*}(\cdot, \pi)\|_\infty \geq 0$, which measures the value distance of θ to the optimal parameter at some policy π. In the following, we analyze the generalization performance on value estimation of unlearned policy π_2 when PeVFA $\mathbb{V}_\theta$ is only trained to approximate the values of policy π_1. We first introduce the two assumptions below:

Assumption 1 (Contraction)**.** *The learning process $\mathscr{P}$ of value approximation for π_1 is γ-contraction, that is $f_{\hat{\theta}}(\pi_1) \leq \gamma f_\theta(\pi_1)$ where $\gamma \in (0, 1]$ and $\theta \xrightarrow{\mathscr{P}(\pi_1)} \hat{\theta}$.*

θ and $\hat{\theta}$ are the parameters of PeVFA before and after some learning of π_1's values. We use f and $\hat{f}$ as abbreviations for f_θ and $f_{\hat{\theta}}$ below for the ease of demonstration. Recent works (Keskar et al., 2017; Novak et al., 2018; Wang et al., 2018) suggest that generalization performance is related to local properties of the model. We then assume following smoothness property for following analysis:

Assumption 2 (Smoothness)**.** *$\hat{f}$ is (or has) Lipschitz continuous (gradient/Hessian) at π_1 with Lipschitz constant $\hat{L}_0$, e.g., $|\hat{f}(\pi) - \hat{f}(\pi_1)| \leq \hat{L}_0 \cdot d(\pi, \pi_1)$ for $\pi \in \Pi$ with some metric space (Π, d).*

With above two assumptions, next we derive a general form of an upper bound of the generalized value approximation for unlearned policy π_2 (i.e., $\hat{f}(\pi_2)$)) as follows:

Theorem 1. *Under Assumption 1 and 2, when $f(\pi_1) \leq f(\pi_2)$, we have the following bound:*

$$\hat{f}(\pi_2) \leq \underbrace{\gamma f(\pi_2)}_{\textit{generalized contraction}} + \underbrace{\mathcal{M}(\pi_1, \pi_2, \hat{L})}_{\textit{locality margin}}. \tag{2}$$

The form of $\mathcal{M}$ depends on the smoothness property, e.g., $\mathcal{M}(\pi_1, \pi_2, \hat{L}) = \hat{L}_0 \cdot d(\pi, \pi_1)$ when $\hat{f}$ is Lipschitz continuous. See Appendix A.1 for more instances of $\mathcal{M}$ when higher-order smoothness properties (Nesterov & Polyak, 2006) are considered.

Proof. See Appendix A.1. The main idea is to chain the Lipschitz continuity upper bound (Assumption 2), the contraction upper bound (Assumption 1), and the inequality of f. □

Remark 1. *The case $f(\pi_1) \leq f(\pi_2)$ considered in Theorem 1 can usually exist since only π_1's values are learned. Under such circumstances, we suggest that the complementary case $f(\pi_1) > f(\pi_2)$ is acceptable since the approximation error of π_2 is already lower than the trained one.*

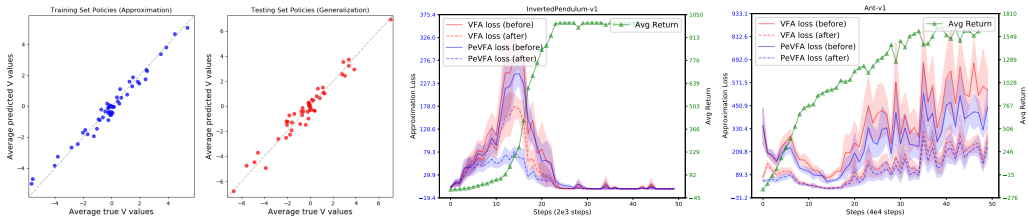
Theorem 1 provides a generalization upper bound for $\hat{f}(\pi_2)$, as a generalized contraction on $f(\pi_2)$ plus a locality margin term $\mathcal{M}$ which is related to π_1, π_2 and the smoothness property $\hat{L}$. Further, we analyze the condition when a PeVFA can also obtain a contraction approximation for unlearned policy π_2 though only trained on π_1, as below:

Corollary 1. *Followed by Theorem 1, consider $\hat{f}$ is Lipschitz continuous and $f(\pi_2) \neq 0$, we have, $\hat{f}(\pi_2) \leq \gamma_g f(\pi_2)$ where $\gamma_g = \gamma + \frac{\hat{L}_0 \cdot d(\pi_1, \pi_2)}{f(\pi_2)}$. When $f(\pi_2) \geq \frac{\hat{L}_0 \cdot d(\pi_1, \pi_2)}{1-\gamma}$, $\theta \xrightarrow{\mathscr{P}(\pi_1)} \hat{\theta}$ is also a γ_g-contraction for value approximation of π_2, with $\gamma_g \in (0, 1]$.*

Proof. Replace $\mathcal{M}$ with Lipschitz continuous upper bound and transform the RHS in Equation 2, then let RHS not greater than $f(\pi_2)$. See Appendix A.2 for complete derivation. □

Remark 2. *From the generalization contraction condition provided in Corollary 1, we can find that: as i. $\gamma \to 0$, or ii. $d(\pi_1, \pi_2) \to 0$, or iii. $\hat{L}_0 \to 0$, the contraction condition is easier to achieve (or the contraction gets tighter), i.e., the generalization on unlearned policy π_2 is better.*

In the above, we discuss value generalization in a two-policy case, i.e., from one learned policy to another unlearned policy. Global generalization in Figure 2(a) is an extension scenario of the two-policy case where values generalize from a known policy set ($\pi \in \Pi_0$) of which the values are learned to unseen policies ($\pi' \in \Pi_1$). In this paper, we focus more on local generalization of PeVFA as shown in Figure 2(b), Recall the GPI form shown in the left of Figure 1, at each iteration t, value



(a) Global generalization (b) Local generalization

Figure 3: Empirical evidences of two kinds of generalization of PeVFA. (a) *Global generalization*: PeVFA shows comparable value estimation performance on testing policy set (red) while maintains reasonable order of optimality. (b) *Local generalization*: PeVFA ($\mathbb{V}_\theta(\chi_\pi)$) shows lower losses (i.e., closer distance to approximation target) than convention VFA (V_ϕ^π) before and after the training for values of successive policies along policy improvement path, which demonstrates Corollary 2.

generalization from current policy π_t to improved policy π_{t+1} is exactly the two-policy case we discussed above. However, it is unclear how local generalization can impact the value estimation in GPI (i,e., policy evaluation). To this end, we propose the following Corollary to see a connection between local generalization of PeVFA and a closer value approximation distance:

Corollary 2. *At iteration t in local generalization scenario of PeVFA (Figure 2(b)), if $f_{\theta_t}(\pi_t) + f_{\theta_t}(\pi_{t+1}) \leq \|\mathbb{V}_{\theta^*}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\|$, then $f_{\theta_t}(\pi_{t+1}) \leq \|\mathbb{V}_{\theta_t}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\|$.*

Proof. Proof can be obtained by applying *Triangle Inequality*. See Appendix A.3. □

Corollary 2 indicates that local generalization of PeVFA can induce a more preferable start point ($\mathbb{V}_{\theta_t}(\pi_{t+1})$) which is closer to the optimal approximation target ($\mathbb{V}_{\theta^*}(\pi_{t+1})$) than the conventional one ($V_{\theta_t}^{\pi_t}$, equivalent to $\mathbb{V}_{\theta_t}(\pi_t)$ in definition) for policy evaluation process at iteration $t+1$.

Remark 3. *With closer start points $\mathbb{V}_{\theta_t}(\pi_{t+1})$, policy evaluation (i.e., minimize $f_{\theta_t(\pi_{t+1})}$) can be more efficient with PeVFA. One can assume an ideal case with perfect local generalization, where policy evaluation is no longer necessary and policy improvement can be performed consecutively.*

3.2 Empirical Evidences for Two Kinds of Generalization

Beyond theoretical discussion, we empirically investigate whether two kinds of generalization can be achieved by a PeVFA neural network. First, we demonstrate global generalization in a continuous 2D Point Walker environment. We build the policy sets Π with synthetic policies, each of which is a randomly initialized 2-layer *tanh*-activated neural policy network with 2 units for each layer. Here we use the concatenation of all weights and biases of the policy network as representation χ_π for each policy π for demonstration, called *raw* policy representation (RPR), while more advanced policy representation methods will be introduced in Sec. 4. We rollout the policies in the environment to collect trajectories $\mathcal{T} = \{\tau_i\}_{i=0}^k$ and then obtain the dataset $\{(\chi_{\pi_j}, \mathcal{T}_{\pi_j})\}_{j=0}^n$. We separate the synthetic policy set into training set (i.e., known policies Π_0) and testing set (i.e., unseen policies Π_1). Figure 3(a) shows that a PeVFA trained with data collected by Π_0 achieves comparable testing approximation performance when evaluating values of policies in Π_1, as well as maintains the order of optimality among policies. More experimental details are provided in Appendix B.1.

We then demonstrate local generalization of PeVFA to examine the existence of Corollary 2. We also use a 2-layer 2-unit policy network trained by PPO (Schulman et al., 2017) algorithm in OpenAI MuJoCo continuous control tasks. Parallel to the conventional value network V_ϕ^π(s) (i.e., VFA) in PPO, we set a PeVFA network $\mathbb{V}_\theta(s, \chi_\pi)$ with RPR as input. Compared with V_ϕ^π, $\mathbb{V}_\theta(s, \chi_\pi)$ is additionally trained with data from historical policies ($\{\pi_i\}_{i=0}^t$) along the policy improvement path. Note that $\mathbb{V}_\theta(\chi_\pi)$ does not interfere with PPO training here, and is only referred as a comparison with V_ϕ^π on the approximation error to the true values (i.e., collected returns) of successive policy π_{t+1}. The results in Figure 3(b) demonstrate the local generalization of PeVFA in InvertedPendulum-v1 and Ant-v1. See Appendix B.2 for complete results in 7 MuJoCo tasks and more experimental details.

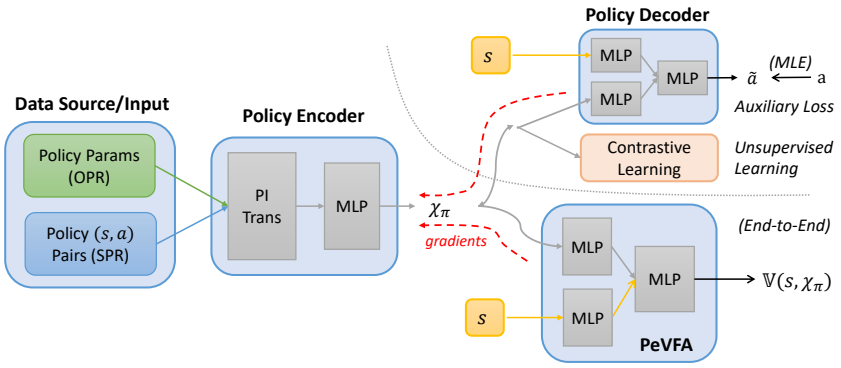


Figure 4: Framework of policy representation training. Policy network parameters (OPR) or policy state-action pairs (SPR) are fed in policy encoder with permutation-invariant (PI) transformations, producing representation χ_π. χ_π can be trained by gradients from PeVFA value approximation (i.e., End-to-End), as well as with optional auxiliary action recovery loss and unsupervised contrastive learning, shown with separation by the gray dotted line.

3.3 Reinforcement Learning with PeVFA

Since PeVFA extends the capacity of conventional VFA and induces value generalization among policies, we expect to derive improved RL algorithms with PeVFA by utilizing the two kinds of generalization we discussed in previous sections. In this paper, we focus on local generalization and propose a new GPI form with PeVFA. As illustrated in the right of Figure 1, the key difference is that the generalized value approximation $\mathbb{V}_{\theta_t}(\chi_{\pi_{t+1}})$ of successive policy π_{t+1} is taken as the start point of policy evaluation at each iteration. One can see the difference by comparing $\mathbb{V}_{\theta_t}(\chi_{\pi_{t+1}})$ with the conventional one $V_{\phi_t}^{\pi_t}$, which is equivalent to $\mathbb{V}_{\theta_t}(\chi_{\pi_t})$ in definition. From Corollary 2 and the evidence in Figure 3(b), such generalized starting points are closer to approximation target (i.e., true values v^π) in some cases, thus we suggest that these local generalization steps (gray arrows) help to reduce approximation error with finite updates and improve the efficiency of the overall learning process. See a pseudo-code of GPI with PeVFA (Algorithm 2) and more discussion in Appendix C.

For global generalization, we assume that one can obtain a high-quality evaluation of unseen policies by training a PeVFA only on known policies. This indicates an appealing potential to circumvent extra sample collection and training cost for policy evaluation in many problems, e.g., evolutionary strategy (Salimans et al., 2017), opponent modeling (He & Boyd-Graber, 2016) and policy transfer and adaptation (Arnekvist et al., 2019; Raileanu et al., 2020), which beyond the scope of this paper.

4 Policy Representation Learning

In last section, we analyze value generalization among policies induced by PeVFA and propose a new GPI for more efficient learning process. To derive practical deep RL algorithms, one key problem is policy representation, i.e., a low-dimensional embedding of RL policy. To our knowledge, it remains unclear how to obtain an effective and compact representation for an RL policy.

We propose a framework of policy representation learning as shown in Figure 4. Our intuitions come from answering the question: what makes a good policy representation, particularly when the policy is a neural network. Our proposed policy representations are motivated by a geometric perspective of the policy as a curved surface of the policy distribution; details are provided in the Appendix D.1. Concretely, we provide two policy representations below:

- *Surface Policy Representation (SPR)*: The first way is to extract policy representation from the state-action pairs (or trajectories), since they reflect the information about how policy may behave under such states. We view this as scattering sample points on policy curved surface, which should be able to capture the features as the number of samples increases.
- *Origin Policy Representation (OPR)*: Moreover, we propose a novel way to learn a low-dimensional embedding from the policy network parameters directly. Generally, we consider a policy network to be an MLP with well represented state features (e.g., features

extracted by CNN for pixels or by LSTM for sequences) as input and deterministic or stochastic policy output. We then define a lossy mapping to compress all the weights and biases of the MLP to obtain the corresponding OPR.

For both SPR and OPR, we use permutation-invariant transformations, i.e., mainly use MLP then Mean-Reduce operation, and detailed implementation is provided in Appendix D.2. A few works also involve representation or embedding learning for RL policy in Multiagent Learning (Grover et al., 2018), Hierarchical RL (Wang et al., 2020), and Policy Transfer (Raileanu et al., 2020; Hausman et al., 2018). We found almost all of them belongs to the scope of SPR and a detailed review for them is provided in Appendix D.5. For OPR, our goal is to learn a representation of the policy itself from its parameters rather than of the architecture of a policy network as is common in Neural Architecture Search (NAS) (Zhou et al., 2019; Ma et al., 2020).

The most straightforward way to train policy representation χ_π together with PeVFA end-to-end as shown in Figure 4 by backpropagating the value approximation error through the policy representation. Intuitively, a good representation should be able to predict the policy's behavior. To this end, we resort to an auxiliary loss to recover the actions of π from χ_π under different states (Grover et al., 2018; Raileanu et al., 2020). Concretely, an auxiliary policy decoder (or a master policy) $\bar{\pi}(\cdot|s, \chi_\pi)$ is trained through Maximum Likelihood Estimation (MLE), i.e., to minimize $-\sum_{(s,a)\sim\mathcal{T}_\pi} \log \bar{\pi}(a|s, \chi_\pi)$. In addition, we propose to improve the representation learning of χ_π with unsupervised Contrastive Learning (Oord et al., 2018; Srinivas et al., 2020). In this way, policies are encouraged to be close to similar ones (i.e., positive samples) and to be apart from different ones (i.e., negative samples) in policy representation space. For each policy, we construct positive samples by data augmentation on policy data, depending on SPR or OPR considered; and different policies along the policy improvement path naturally provides negative samples for each other. The policy representation network is then trained with InfoNCE loss (Oord et al., 2018) as commonly done in (He et al., 2020; Chen et al., 2020). More implementation details are in Appendix D.

5 EXPERIMENTS

We conduct several experiments with our proposed methods and try to answer the following questions: *i.* How much does the local generalization property of PeVFA benefit a deep RL learning algorithm? *ii.* Is our policy representation learning framework effective to learn good representation for a policy network?

To answer above questions, we derive a practical deep RL algorithm with PeVFA based on Proximal Policy Optimization (PPO) (Schulman et al., 2017) to demonstrate and evaluate GPI with PeVFA. PPO is a policy optimization algorithm that learns both a policy network π and a value network V_ϕ that approximates the values of π, typically following GPI in the left of Figure 1. The policy π is optimized with respect to a surrogate objective (Schulman et al., 2015) using advantages calculated by V_ϕ and GAE (Schulman et al., 2016). Due to the popularity and simplicity in implementation, we use PPO as the base algorithm for a representative implementation of GPI with PeVFA. We propose to replace the conventional value network $V_\phi(s)$ by a PeVFA network $\mathbb{V}_\theta(s, \chi_\pi)$ with parameters θ as given in Figure 4, and perform GPI with PeVFA as in the right of Figure 1. Without modifying the policy optimization process of vanilla PPO, we then obtain a new algorithm, called PPO-PeVFA. Note that what is actually changed here is simply a drop-and-replacement of the value function.

Training. The only difference of PPO-PeVFA is the training of policy representation χ_π and PeVFA network $\mathbb{V}_\theta(s, \chi_\pi)$. For policy representation χ_π, it depends on policy data used and training losses adopted as introduced in Sec. 4. See Algorithm 3 in Appendix D.4 for an overall pseudo-code of policy representation training. For PeVFA network $\mathbb{V}_\theta(s, \chi_\pi)$, it is trained to approximate the values of multiple policies along the policy improvement (i.e. optimization) path, as Algorithm 2 in Appendix C. Both of above training use historical policies, whose data (i.e., network parameters and state-action pairs) are stored. Note that for PPO (and PPO-PeVFA), new policies are only produced after policy optimization process. We do not assume access to pre-collected policies and thus the size of policy set increases from 1 during the learning process. In our experiments, about 1k to 2k policies are produced in an experimental trial. We use all historical policies in our training and for the case with many more policies, other sophisticated training methods can be considered.

Results. We conduct the following experiments in OpenAI Gym MuJoCo continuous control tasks (Brockman et al., 2016; Todorov et al., 2012). Beside the comparison to PPO, we consider an-

Table 1: Max Average Return over 10 trials of 1M time steps (2M for Ant-v1). First two maximum values for each task are bolded. $\pm$ corresponds to half a std over trials.

	Benchmarks		Origin Policy Representation (Ours)			Surface Policy Representation (Ours)		
Environments	PPO	Ran PR	E2E	CL	AUX	E2E	CL	AUX
HalfCheetah-v1	2621	2470	3171 $\pm$ 427.63	**3725** $\pm$ **348.55**	3175 $\pm$ 517.52	2774 $\pm$ 233.39	**3349** $\pm$ **341.42**	3216 $\pm$ 506.39
Hopper-v1	1639	1226	2085 $\pm$ 310.91	2351 $\pm$ 231.11	2214 $\pm$ 360.78	2227 $\pm$ 297.35	**2392** $\pm$ **263.93**	**2577** $\pm$ **217.73**
Walker2d-v1	1505	1269	1856 $\pm$ 305.51	2038 $\pm$ 315.51	**2044** $\pm$ **316.32**	1930.57 $\pm$ 456.02	**2203** $\pm$ **381.95**	1980 $\pm$ 325.54
Ant-v1	2835	2742	3581 $\pm$ 185.43	**4019** $\pm$ **162.47**	**3784** $\pm$ **268.99**	3173 $\pm$ 184.75	3632 $\pm$ 134.27	3397 $\pm$ 200.03
InvDouPend-v1	9344	9355	**9357** $\pm$ **0.29**	9355 $\pm$ 0.64	9355 $\pm$ 0.68	9355 $\pm$ 0.89	**9356** $\pm$ **0.96**	9355 $\pm$ 1.42
LunarLander-v2	219	226	**238** $\pm$ **3.37**	**239** $\pm$ **3.70**	234 $\pm$ 3.47	236 $\pm$ 3.13	234 $\pm$ 3.13	235 $\pm$ 5.70

other benchmark, PPO-PeVFA with fixed randomly generated policy representation for each policy, namely Ran PR. The overall experimental results are in Table 1. Exact experimental details and complete learning curves are provided in Appendix F due to space limitation.

Question 1. *Can PPO-PeVFA with only End-to-End (E2E) policy representation outperform PPO?*

From Table 1, we can find that both PPO-PeVFA with OPR and with SPR trained in E2E fashion outperforms PPO in all 6 tasks, especially in Ant-v1. This indicates that effective policy representation can be learned from value approximation loss in our experiments, based on which PeVFA can benefit from local generalization thus improve the learning process. Additionally, Ran PR shows an overall degeneration compared to PPO, demonstrating the significance of OPR and SPR.

Question 2. *Can representation learned through contrastive learning (CL) and auxiliary loss (AUX) further benefit performance of PPO-PeVFA?*

For both OPR and SPR, we observe consistent improvements induced by CL and AUX over E2E, which means that additional learning to maintain unsupervised consistency (CL) and capture policy behavior (AUX) further benefits policy representation, and eventually results in substantial improvement over PPO in all tasks except for the first two simple tasks. In an overall view, OPR slightly outperforms SPR and as CL does over AUX. We hypothesize that it is due to the stochasticity of state-action pairs which serve as inputs of SPR and training samples for AUX. We suggest that this remains space for future improvement.

Question 3. *How is the learned representation like when viewing in a low-dimensional space?*

We show a 2D t-SNE (Maaten & Hinton, 2008) view in Figure 5 and more visualisations in Appendix F. Policies from different trials are locally continuous (multimodality) while a global evolvement emerges with respect to policy performance.

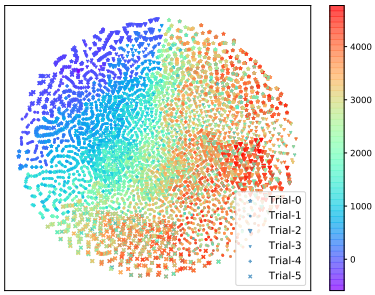


Figure 5: t-SNE visualization for OPR (E2E) of 6k policies from 5 trials (denoted by different markers) in Ant-v1. Policies are colored by average return.

6 Conclusion and Future Work

In this paper, we propose Policy-extended Value Function Approximator (PeVFA) which induces value generalization among policies. We propose a new form of GPI with PeVFA which can potentially benefit from local value generalization along the policy improvement path. Moreover, we propose several approaches to learn an effective low-dimensional representation of an RL policy. Our experiments demonstrate the effectiveness of the generalization properties of PeVFA and our proposed methods of representing the RL policy.

Our work opens up new ways to potentially improve the policy learning process from global or local perspectives using PeVFAs that receive an explicit policy representation to improve generalization of the value approximation. We believe our work can guide future research directions on policy representations and PeVFA in many RL problems. We plan to further study policy representation learning, as well as more complex value generalization scenarios, including TD learning with non-stationary approximation targets. In addition, optimizing policies in the representation space can also be an interesting direction.

REFERENCES

M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. In *NeurIPS*, pp. 5048–5058, 2017.

I. Arnekvist, D. Kragic, and J. A. Stork. VPE: variational policy embedding for transfer reinforcement learning. In *ICRA*, pp. 36–42, 2019.

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *ICML*, volume 70, pp. 449–458, 2017.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709, 2020.

H. Fu, H. Tang, J. Hao, C. Chen, X. Feng, D. Li, and W. Liu. Towards effective context for meta-reinforcement learning: an approach based on contrastive learning. *CoRR*, abs/2009.13891, 2020.

S. Fujimoto, H. v. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *ICML*, 2018.

A. Grover, M. Al-Shedivat, J. K. Gupta, Y. Burda, and H. Edwards. Learning policy representations in multiagent systems. In *ICML*, volume 80, pp. 1797–1806, 2018.

D. Hafner, T. P. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *ICLR*, 2020.

K. Hausman, J. Tobias Springenberg, Z. Wang, N. Heess, and M. A. Riedmiller. Learning an embedding space for transferable robot skills. In *ICLR*, 2018.

H. He and J. L. Boyd-Graber. Opponent modeling in deep reinforcement learning. In *ICML*, volume 48, pp. 1804–1813, 2016.

K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pp. 9726–9735, 2020.

N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *CoRR*, abs/2004.14990, 2020.

K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. *CoRR*, abs/2005.06800, 2020.

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2015.

H. Liu, K. Simonyan, and Y. Yang. DARTS: differentiable architecture search. In *ICLR*, 2019.

R. Luo, F. Tian, T. Qin, E. Chen, and T. Liu. Neural architecture optimization. In *NeurIPS*, pp. 7827–7838, 2018.

X. Ma, J. K. Gupta, and M. J. Kochenderfer. Normalizing flow model for policy representation in continuous action multi-agent systems. In *AAMAS*, pp. 1916–1918, 2020.

L. V. D. Maaten and G. E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

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

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

O. Nachum, S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *CoRR*, abs/1805.08296, 2018.

Y. E. Nesterov and B. T. Polyak. Cubic regularization of newton method and its global performance. *Math. Program.*, 108(1):177–205, 2006.

R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein. Sensitivity and generalization in neural networks: an empirical study. In *ICLR*, 2018.

A. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

R. Raileanu, M. Goldstein, A. Szlam, and R. Fergus. Fast adaptation via policy-dynamics value functions. *CoRR*, abs/2007.02879, 2020.

K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, volume 97, pp. 5331–5340, 2019.

T. Salimans, J. Ho, X. Chen, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017.

T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *ICML*, volume 37, pp. 1312–1320, 2015.

J. S Schreck, C. W Coley, and K. JM Bishop. Learning retrosynthetic planning through simulated experience. *ACS central science*, 5(6):970–981, 2019.

J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz. Trust region policy optimization. In *ICML*, pp. 1889–1897, 2015.

J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *ICLR*, 2016.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, pp. 387–395, 2014.

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

A. Srinivas, M. Laskin, and P. Abbeel. CURL: contrastive unsupervised representations for reinforcement learning. *CoRR*, abs/2004.04136, 2020.

R. S. Sutton and A. G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998.

R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup. Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *AAMAS*, pp. 761–768, 2011.

A. Tacchetti, H. F. Song, P. A. M. Mediano, V. Flores Zambaldi, J. Kramár, N. C. Rabinowitz, Th. Graepel, M. Botvinick, and P. W. Battaglia. Relational forward models for multi-agent learning. In *ICLR*, 2019.

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

H. v. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (eds.), *NeurIPS*, pp. 2613–2621, 2010.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

H. Wang, N. S. Keskar, C. Xiong, and R. Socher. Identifying generalization properties in neural networks. *CoRR*, abs/1809.07402, 2018.

R. Wang, R. Yu, B. An, and Z. Rabinovich. I^2hrl: Interactive influence-based hierarchical reinforcement learning. In *IJCAI*, pp. 3131–3138, 2020.

J. You, B. Liu, Z. Ying, V. S. Pande, and J. Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *NeurIPS 2018*, pp. 6412–6422, 2018.

W. Zhou, Y. Yu, Y. Chen, K. Guan, T. Lv, C. Fan, and Z. Zhou. Reinforcement learning experience reuse with policy residual representation. In *IJCAI*, pp. 4447–4453, 2019.

A PROOFS

A.1 PROOF OF THEOREM 1

Proof. Start from Assumption 2, we derive the upper bound of $\hat{f}(\pi_2)$:

$$\hat{f}(\pi_2) \leq \hat{f}(\pi_1) + \mathcal{M}(\pi_1, \pi_2, \hat{L}), \tag{3}$$

where $\mathcal{M}(\pi_1, \pi_2, \hat{L})$ can have different forms that depends on the specific smoothness property, for examples:

$$\mathcal{M}(\pi_1, \pi_2, \hat{L}) = \begin{cases} \hat{L}_0 \cdot d(\pi_1, \pi_2) & ① \\ \langle \hat{f}'(\pi_1), \pi_2 - \pi_1 \rangle + \frac{1}{2}\hat{L}_1 \cdot d(\pi_1, \pi_2)^2 & ② \\ \langle \hat{f}'(\pi_1), \pi_2 - \pi_1 \rangle + \frac{1}{2}\langle \hat{f}''(\pi_1)(\pi_2 - \pi_1), \pi_2 - \pi_1 \rangle + \frac{1}{6}\hat{L}_2 \cdot d(\pi_1, \pi_2)^3 & ③ \end{cases}$$

①, ②, ③ correspond to Lipschitz Continuous, Lipschitz Gradients and Lipschitz Hessian (Nesterov & Polyak, 2006).

Combined with Assumption 1 and consider the case $f(\pi_1) \leq f(\pi_2)$, Equation 3 can be further transformed as follows:

$$\begin{aligned} \hat{f}(\pi_2) &\leq \hat{f}(\pi_1) + \mathcal{M}(\pi_1, \pi_2, \hat{L}) \\ &\leq \gamma f(\pi_1) + \mathcal{M}(\pi_1, \pi_2, \hat{L}) \\ &\leq \underbrace{\gamma f(\pi_2)}_{\text{generalized contraction}} + \underbrace{\mathcal{M}(\pi_1, \pi_2, \hat{L})}_{\text{locality margin}}, \end{aligned} \tag{4}$$

which yields the generalization upper bound in Theorem 1. We note the first term of RHS of Equation 4 as *generalized contraction* since it is from the contraction on $f(\pi_1)$, and the second term as *locality margin* since it is determined by specific local property. □

A.2 PROOF OF COROLLARY 1

Proof. Consider the concrete form of locality margin $\mathcal{M}(\pi_1, \pi_2, \hat{L})$ in the case of Lipschitz continuous, i.e.,

$$\hat{f}(\pi_2) \leq \gamma f(\pi_2) + \hat{L}_0 \cdot d(\pi_1, \pi_2), \tag{5}$$

and consider $f(\pi_2) \neq 0$ (note that $f(\pi_2) \geq 0$):

$$\hat{f}(\pi_2) \leq \gamma_g f(\pi_2), \text{ where } \gamma_g = \gamma + \frac{\hat{L}_0 \cdot d(\pi_1, \pi_2)}{f(\pi_2)}. \tag{6}$$

Then we get the contraction condition of value generalization on π_2 in Corollary 1, by letting the RHS of Equation 5 not greater than $f(\pi_2)$, i.e., $\gamma_g \leq 1$ (note that $\gamma \in (0, 1]$):

$$\begin{aligned} \gamma f(\pi_2) + \hat{L}_0 \cdot d(\pi_1, \pi_2) &\leq f(\pi_2) \\ (1 - \gamma) f(\pi_2) &\geq \hat{L}_0 \cdot d(\pi_1, \pi_2) \\ f(\pi_2) &\geq \frac{\hat{L}_0 \cdot d(\pi_1, \pi_2)}{1 - \gamma}. \end{aligned} \tag{7}$$

Intuitions observed from Equation 7 are discussed in Remark 3. In another word, the tighter the contraction on learned policy π_1 is, the closer the two policies are, the smoother the approximation loss function $\hat{f}$ is, the generalization on unlearned policy π_2 is better. □

Corollary 1 provides the generalization contraction condition on $f(\pi_2)$. We also provide another view from the perspective of $f(\pi_1)$, by additionally considering the smoothness property of f.

Similar as in Assumption 2, we assume that f is Lipschitz continuous at π_1 with Lipschitz constant L_0, e.g., $|f(\pi) - f(\pi_1)| \leq L_0 \cdot d(\pi, \pi_1)$ with some metric space (Π, d). Higher-order smoothness property can also be considered here. We then obtain a lower bound of $f(\pi_2)$ below:

$$f(\pi_1) - L_0 \cdot d(\pi, \pi_1) \leq f(\pi_2). \tag{8}$$

We then chain the second line of RHS in Equation 4 ($\mathcal{M}$ in the case of Lipschitz continuous) and the LHS of Equation 8:

$$\hat{f}(\pi_2) \leq \gamma f(\pi_1) + \hat{L}_0 \cdot d(\pi, \pi_1) \leq f(\pi_1) - L_0 \cdot d(\pi, \pi_1) \leq f(\pi_2), \tag{9}$$

re-arrange and yield the generalization contraction condition on $f(\pi_1)$:

$$\begin{aligned} \gamma f(\pi_1) + \hat{L}_0 \cdot d(\pi 1, \pi_2) &\leq f(\pi_1) - L_0 \cdot d(\pi_1, \pi_2) \\ (1-\gamma) f(\pi_1) &\geq (L_0 + \hat{L}_0) d(\pi_1, \pi_2) \\ f(\pi_1) &\geq \frac{(L_0 + \hat{L}_0) d(\pi_1, \pi_2)}{1-\gamma}. \end{aligned} \tag{10}$$

A.3 Proof of Corollary 2

Proof. Due to *Triangle Inequality*, we have:

$$\begin{aligned} f_{\theta_t}(\pi_t) + \|\mathbb{V}_{\theta_t}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\| &\geq \|\mathbb{V}_{\theta^*}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\| \\ \text{i.e., } \|\mathbb{V}_{\theta_t}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_t)\| + \|\mathbb{V}_{\theta_t}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\| &\geq \|\mathbb{V}_{\theta^*}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\| \end{aligned} \tag{11}$$

Combined with condition

$$\begin{aligned} f_{\theta_t}(\pi_t) + f_{\theta_t}(\pi_{t+1}) &\leq \|\mathbb{V}_{\theta^*}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\| \\ \text{i.e., } \|\mathbb{V}_{\theta_t}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_t)\| + \|\mathbb{V}_{\theta_t}(\pi_{t+1}) - \mathbb{V}_{\theta^*}(\pi_{t+1})\| &\leq \|\mathbb{V}_{\theta^*}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\|. \end{aligned} \tag{12}$$

Chain above two inequality,

$$\begin{aligned} f_{\theta_t}(\pi_t) + f_{\theta_t}(\pi_{t+1}) \leq \|\mathbb{V}_{\theta^*}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\| &\leq f_{\theta_t}(\pi_t) + \|\mathbb{V}_{\theta_t}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\| \\ \underbrace{f_{\theta_t}(\pi_{t+1})}_{\text{generalizated VAD with PeVFA}} &\leq \underbrace{\|\mathbb{V}_{\theta_t}(\pi_t) - \mathbb{V}_{\theta^*}(\pi_{t+1})\|}_{\text{conventional VAD}}, \end{aligned} \tag{13}$$

then we have that with local generalization of values for successive policy π_{t+1}, the value approximation distance (VAD) can be closer in contrast to the conventional one (RHS of Equation 13). □

B Details of Empirical Evidence of Two Kinds of Generalization

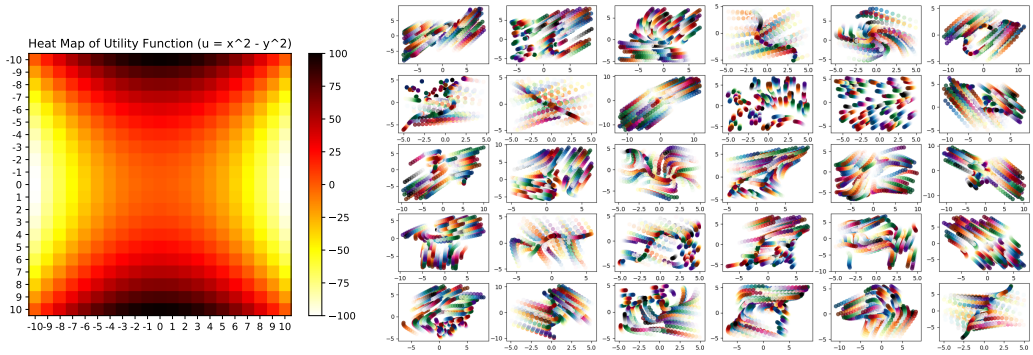
B.1 Global Generalization in 2D Point Walker

We consider a point walker on a 2D continuous plane with

- state: $(x, y, \sin(\theta), \cos(\theta), \cos(x), \cos(y))$, where θ is the angle of the polar coordinates,
- action: 2D displacement, $a \in \mathbb{R}^2_{[-1,1]}$,
- a deterministic transition function xxx,
- a reward function: $r_t = \frac{u_{t+1} - u_t}{10}$ with utility $u_t = x_t^2 - y_t^2$, as illustrated in Figure 6(a).

Global generalization denotes the generalization scenario that values can generalize to unlearned policies ($\pi' \in \Pi_1$) from already learned policies ($\pi \in \Pi_0$). We build the policy sets $\Pi = \Pi_0 \cup \Pi_1$ and $\Pi_0 \cap \Pi_1 = \emptyset$ with synthetic policies. Each synthetic policy is a 2-layer *tanh*-activated neural policy network with random initialization. Each policy is deterministic, taking an environmental state as input and outputting a displacement in the plane. We find that the synthetic population generated by such a simple way can show diverse behaviors. Figure 6(b) shows the motion patterns of an example of such a synthetic population. Note that the synthetic policies are not trained in this experiment.

We use such small policy networks for the convenience of acquisition of policy representation. Here we use the concatenation of all weights and biases of the policy network (22 in total) as representation χ_π for each policy π, called *raw* policy representation (RPR). We rollout the policies in environment to collect trajectories $\mathcal{T} = \{\tau_i\}_{i=0}^k$ and then obtain the dataset $\{(\chi_{\pi_j}, \mathcal{T}_{\pi_j})\}_{j=0}^n$. We separate the policy set into training set (i.e., unknown policies Π_0) and testing set (i.e., unseen policies Π_1). We set a PeVFA network $\mathbb{V}_\theta(\chi_\pi)$ to approximate the values of training policies (i.e., $\pi \in \Pi_0$), and then we evaluate the values on testing policies (i.e., $\pi \in \Pi_1$)



(a) Utility function heat map

(b) Examples of synthetic policy population

Figure 6: 2D Pointer Walker. (a) The heat map of the utility function of the 2D plane. The darker regions have higher utilities. (b) Demonstrative illustrations of trajectories generated by 30 synthetic policies, showing diverse behaviors and patterns. Each subplot illustrates the trajectories generated in 50 episodes by a randomly synthetic policy, with different colors as separation. For each trajectory (the same color in one subplot), transparency represents the dynamics along timesteps, i.e., fully transparent and non-transparent denotes the positions at first and last timesteps.

B.2 Local Generalization in MuJoCo Continuous Control Tasks

We demonstrate local generalization of PeVFA, especially to examine the existence of Corollary 2. We use a 2-layer 2-unit policy network trained by PPO (Schulman et al., 2017) algorithm in OpenAI MuJoCo continuous control tasks. Using a very small policy network is for the convenience of training and acquisition of policy representation. We use all weights and biases of the small policy network (also called *raw* policy representation, RPR), whose number is about 10 to 100 in our experiments, depending on the specific environment (i.e., the state and action dimensions).

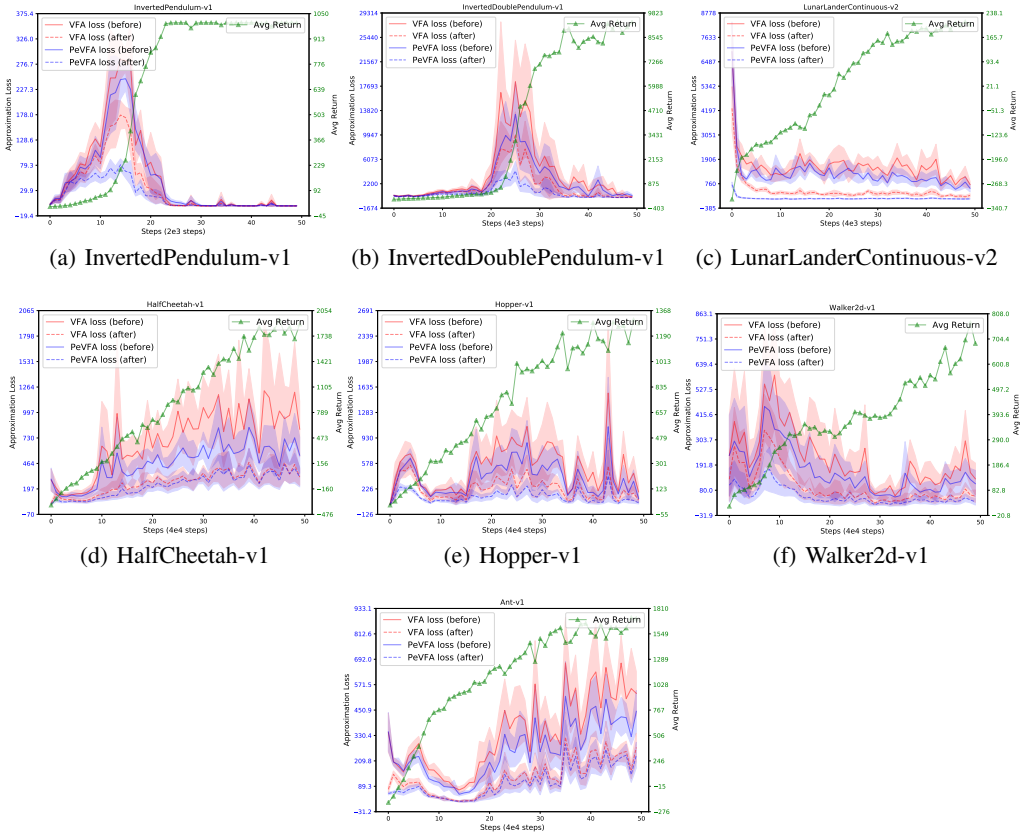
We train the small policy network as commonly done with PPO and GAE (Schulman et al., 2016). The convention value network V_ϕ^π(s) (VFA), is a 2-layer 128-unit ReLU-activated MLP with state as input and value as output. Parallel to the conventional VFA in PPO, we set a PeVFA network $\mathbb{V}_\theta(s, \chi_\pi)$ with RPR as additional input. The structure of PeVFA differs at the first hidden layer which has two input streams and each of them has 64 units, so that make VFA and PeVFA have similar scales of parameter number. Compared with V_ϕ^π, $\mathbb{V}_\theta(s, \chi_\pi)$ is additionally trained with data from historical policies ($\{\pi_i\}_{i=0}^t$) along the policy improvement path (e.g., Algorithm 2).

Note that $\mathbb{V}_\theta(\chi_\pi)$ does not interfere with PPO training here, and is only referred as a comparison with V_ϕ^π on the approximation error to the true values of successive policy π_{t+1}. We use the MC returns of on-policy data (i.e., trajectories) collected by current successive policy as unbiased estimates of true values, similarly done in (v. Hasselt, 2010). Then we calculate the approximation error of VFA and PeVFA to the approximation target before and after value network training, which can be done average the value network losses among training batches. Finally, we compare the approximation error between VFA and PeVFA to approximately examine local generalization and closer approximation target in Corollary 2. Complete results of local generalization across all 7 MuJoCo tasks are show in Figure 7.

C Generalized Policy Iteration with PeVFA

C.1 Comparison between Conventional GPI and GPI with PeVFA

From the lens of Generalized Policy Iteration (Sutton & Barto, 1998), for most model-free policy-based RL algorithms, the approximation of value function and the update of policy through policy gradient theorem are usually conducted iteratively. Representative examples are REINFORCE (Sutton & Barto, 1998), Advantage Acotor-Critic (Mnih et al., 2016), Deterministic Policy Gradient (DPG) (Silver et al., 2014) and Proximal Policy Optimization (PPO) (Schulman et al., 2017). With conventional value function (approximator), policy evaluation is usually performed in an on-policy



(a) InvertedPendulum-v1 (b) InvertedDoublePendulum-v1 (c) LunarLanderContinuous-v2

(d) HalfCheetah-v1 (e) Hopper-v1 (f) Walker2d-v1

(g) Ant-v1

Figure 7: Complete empirical evidence of local generalization of PeVFA across 7 MuJoCo tasks. Each plot has two vertical axes, the left one for approximation error (red and blue curves) and the right one for average return (green curves). Red and blue denotes the approximation error of conventional VFA ($V_\phi^\pi(s)$) and of PeVFA ($\mathbb{V}_\theta(s, \chi_\pi)$) respectively; solid and dashed curves denote the approximation error before and after the training for values of successive policy (i.e., policy evaluation) with conventional VFA and PeVFA, averaged over 6 trials. The shaded region denotes half a standard deviation of average evaluation. PeVFA consistently shows lower losses (i.e., closer to approximation target) across all tasks than convention VFA before and after policy evaluation along policy improvement path, which demonstrates Corollary 2.

or off-policy fashion. We provide a general GPI description of model-free policy-based RL algorithm with conventional value functions in Algorithm 1.

Note that we use subscript $t-1 \rightarrow t$ (Line 12 in Algorithm 1) to let the updated value functions to correspond to the evaluated policy π_t during policy evaluation process in current iteration.

As a comparison, a new form of GPI with PeVFA is shown in Algorithm 2. Except for the different parameterization of value function, PeVFA can perform additionally training on historical policy experiences at each iteration (Line 7-8). This is naturally compatible with PeVFA since it develops the capacity of conventional value function to preserve the values of multiple policies. Such a training is to improve the value generalization of PeVFA among a policy set or policy space. Note that for value approximation of current policy π_t (Line 10-14), the start points are generalized values of π_t from historical approximation, i.e., $\mathbb{V}_{t-1}(s, \chi_{\pi_t})$ and $\mathbb{Q}_{t-1}(s, a, \chi_{\pi_t})$. In another word, this is the place where local generalization steps (illustrated in Figure 2(b)) are. One may compare with conventional start points ($V_{t-1}^\pi(s)$ and $Q_{t-1}^\pi(s, a)$, Line 12 in Algorithm 1) and see the difference, e.g., $V_{t-1}^\pi(s) \Leftrightarrow V^{\pi_{t-1}}(s) \Leftrightarrow \mathbb{V}_{t-1}(s, \chi_{\pi_{t-1}})$ is different with $\mathbb{V}_{t-1}(s, \chi_{\pi_t})$, where $\Leftrightarrow$ is used to denote an equivalence in definition. As discussed in Sec. 3.2 and 3.3, we suggest that such local

Algorithm 1 Generalized policy iteration for model-free policy-based RL algorithm with conventional value functions ($V^\pi(s)$ or $Q^\pi(s,a)$)

Initialize policy π_0 and $V^\pi_{-1}(s)$ or $Q^\pi_{-1}(s,a)$
Initialize experience buffer $\mathcal{D}$
for iteration t $= 0, 1, 2, \ldots$ **do**
    Rollout policy π_t in the environment and obtain trajectories $\mathcal{T}_t = \{\tau_i\}_{i=0}^k$
    Add experiences $\mathcal{T}_t$ in buffer $\mathcal{D}$
    if *on-policy update* **then**
        Prepare training samples from rollout trajectories $\mathcal{T}_t$
    else if *off-policy update* **then**
        Prepare training samples by sampling from buffer $\mathcal{D}$
    end if
    Calculate approximation target $\{y_i\}_i$ from training samples (e.g., with MC or TD)
    Update $V^\pi_{t-1}(s)$ or $Q^\pi_{t-1}(s,a)$ with $\{(s_i, y_i)\}_i$ or $\{(s_i, a_i, y_i)\}_i$, i.e., $V^\pi_t \longleftarrow V^\pi_{t-1}$ or $Q^\pi_t \longleftarrow Q^\pi_{t-1}$
    Update policy π_t with regard to $V^\pi_t(s)$ or $Q^\pi_t(s,a)$ through policy gradient theorem, i.e., $\pi_{t+1} \longleftarrow \pi_t$
end for

generalization steps help to reduce approximation error and improve efficiency during the learning process.

Algorithm 2 Generalized policy iteration of model-free policy-based RL algorithm with PeVFAs ($\mathbb{V}(s, \chi_\pi)$ or $\mathbb{Q}(s, a, \chi_\pi)$)

Initialize policy π_0 and PeVFA $\mathbb{V}_{-1}(s, \chi_\pi)$ or $\mathbb{Q}_{-1}(s, a, \chi_\pi)$
Initialize experience buffer $\mathcal{D}$
for iteration t $= 0, 1, 2, \ldots$ **do**
    Rollout policy π_t in the environment and obtain trajectories $\mathcal{T}_t = \{\tau_i\}_{i=0}^k$
    Get the policy representation χ_{π_t} for policy π_t (from policy network parameters or policy rollout experiences)
    Add experiences $(\chi_{\pi_t}, \mathcal{T}_t)$ in buffer $\mathcal{D}$
    # Value approximation training for historical policies $\{\pi_i\}_{i=0}^{t-1}$
    Update PeVFA $\mathbb{V}_{t-1}(s, \chi_{\pi_i})$ or $\mathbb{Q}_{t-1}(s, a, \chi_{\pi_i})$ with all historical policy experiences $\{(\chi_{\pi_i}, \mathcal{T}_i)\}_{i=0}^{t-1}$
    # Conventional value approximation training for current policy π_t
    if *on-policy update* **then**
        Update PeVFA $\mathbb{V}_{t-1}(s, \chi_{\pi_t})$ or $\mathbb{Q}_{t-1}(s, a, \chi_{\pi_t})$ for π_t with on-policy experiences $(\chi_{\pi_t}, \mathcal{T}_t)$
    else if *off-policy update* **then**
        Update PeVFA $\mathbb{V}_{t-1}(s, \chi_{\pi_t})$ or $\mathbb{Q}_{t-1}(s, a, \chi_{\pi_t})$ for π_t with off-policy experiences χ_{π_t} and $\{\mathcal{T}_i\}_{i=0}^t$ from experience buffer $\mathcal{D}$
    end if
    $\mathbb{V}_t \longleftarrow \mathbb{V}_{t-1}$ or $\mathbb{Q}_t \longleftarrow \mathbb{Q}_{t-1}$
    Update policy π_t with regard to $\mathbb{V}_t(s, \chi_{\pi_t})$ or $\mathbb{Q}_t(s, a, \chi_{\pi_i})$ through policy gradient theorem, i.e., $\pi_{t+1} \longleftarrow \pi_t$
end for

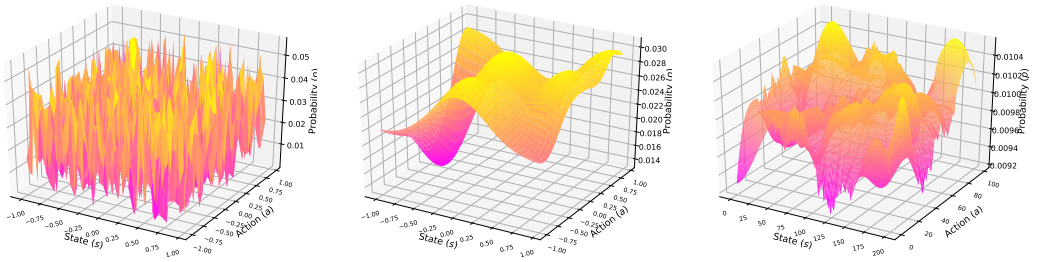
D Policy Representation Learning Details

D.1 Policy Geometry

A policy $\pi \in \Pi = \mathcal{P}(\mathcal{A})^\mathcal{S}$, defines the behavior (action distribution) of the agent under each state. For a more intuitive view, we consider the geometrical shape of a policy: all state $s \in \mathcal{S}$ and all action $a \in \mathcal{A}$ are arranged along the x-axis and y-axis of a 2-dimensional plane, and the probability (density) $\pi(a|s)$ is the value of z-axis over the 2-dimensional plane. Note that for finite state space and finite action space (discrete action space), the policy can be viewed as a $|\mathcal{S}| \times |\mathcal{A}|$ table with each

entry in it is the probability of the corresponding state-action case. Without loss of generality, we consider the continuous state and action space and the policy geometry here. Illustrations of policy geometry examples are shown in Figure 8.

Figure 8(a) shows the policy geometry in a general case, where the policy can be defined arbitrarily. Generally, the policy geometry can be any possible geometrical shape (s.t. $\forall s \in \mathcal{S}, \sum_{a \in \mathcal{A}} \pi(a|s) = 1$). This means that the policy geometry is not necessarily continuous or differentiable in a general case. Specially, one can imagine that the geometry of a deterministic policy consists of peak points ($z = 1$) for each state and other flat regions ($z = 0$). Figure 8(b) shows an example of synthetic continuous policy which can be viewed as a 3D curved surface. In Deep RL, a policy may usually be modeled as a deep neural network. Assume that the neural policy is a function that is almost continuous and differentiable everywhere, the geometry of such a neural policy can also be continuous and differentiable almost everywhere. As shown in Figure 8(c), we provide a demo of neural policy by smoothing an arbitrary policy along both state and action axes.



(a) Arbitrary policy (b) Continuous policy (c) Demo of neural policy

Figure 8: Examples of policy geometry. (a) An arbitrary policy, where $p(s, a) \sim \mathcal{N}(0, 1)$ for each $s \in [-1, 1], a \in [-1, 1]$ and then normalized along action axis. (b) A synthetic continuous policy with $p(s, a) = (1 - a^5 + s^5)\exp(-s^2 - a^2)$ for each $s \in [-1, 1], a \in [-1, 1]$ and then normalized along action axis. (c) A general demo of neural network policy, generated from an arbitrary policy with some smoothing skill.

D.2 Implementation Details of Origin Policy Representation (OPR)

We propose a novel way to learn low-dimensional embedding from policy network parameters directly. To our knowledge, we are the first to learn policy embedding from neural network parameters in RL. Note that latent space of neural networks are also studied in differentiable Network Architecture Search (NAS) (Liu et al., 2019; Luo et al., 2018), where architecture-level embedding are usually considered. In contrast, OPR cares about parameter-level embedding with a given architecture.

Consider a policy network to be a MLP with well-represented state (e.g., CNN for pixels, LSTM for sequences) as input and deterministic or stochastic policy output. We compress all the weights and biases of the MLP to obtain an OPR that represents the decision function. The encoding process of an MLP with two hidden layers is illustrated in Figure 9. The main idea is to perform permutation-invariant transformation for inner-layer weights and biases for each layer first. For each unit of some layer, we view the unit as a non-linear function of all outputs, determined by weights, a bias term and activation function. Thus, the whole layer can be viewed as a batch of operations of previous outputs, e.g., with the shape $[h_t, h_{t-1} + 1]$ for $t \geq 1$ and $t = 0$ is also for the input layer. Note that we neglect activation function in the encoding since we consider the policy network structure is given. That is also why we call OPR as parameter-level embedding in contrast to architecture-level enbedding in NAS (mentioned in last paragraph). We then feed the operation batch in an MLP and perform mean-reduce to outputs. Finally we concatenate encoding of layers and obtain the OPR.

We use permutation-invariant transformation for OPR because that we suggest the operation batch introduced in last paragraph can be permutation-invariant. Actually, our encoding shown in Figure 9 is not strict to obtain permutation-invariant representation since inter-layer dependencies are not included during the encoding process. We also tried to incorporate the order information during

OPR encoding and we found similar results with the way we present in Figure 9, which we adopt in our experiments.

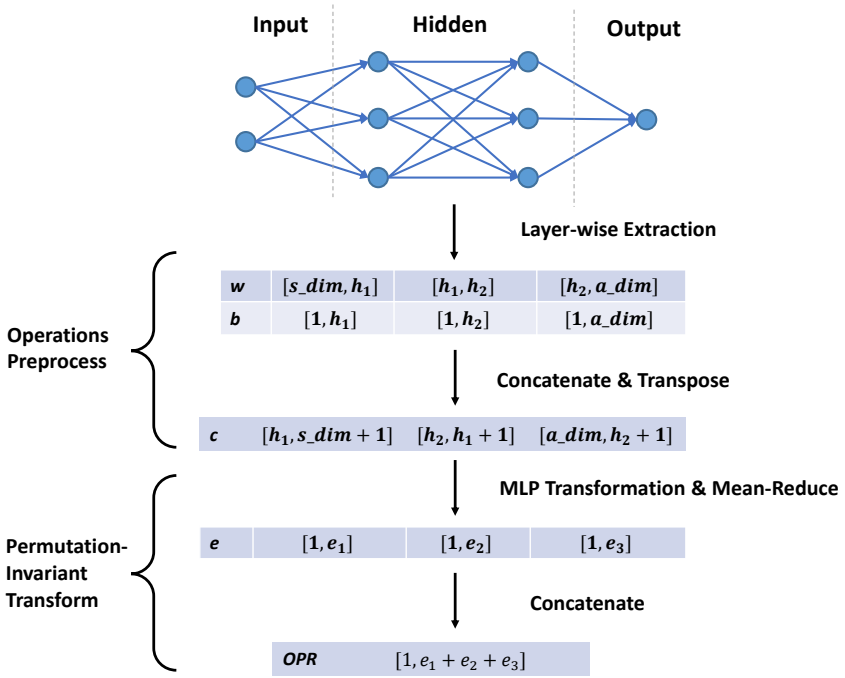


Figure 9: An illustration for policy encoder of Origin Policy Representation (OPR) for a two-layer MLP. h_1, h_2 denotes the numbers of hidden units for the first and second hidden layers respectively. The main idea is to perform permutation-invariant transformation for inner-layer weights and biases for each layer first and then concatenate encoding of layers.

D.3 Data Augmentation for SPR and OPR During Contrastive Learning

Data augmentation is studied to be an important component in contrastive learning in deep RL recently (Laskin et al., 2020). Contrastive learning usually resorts to data augmentation to build postive samples. Data augmentation is typically performed on pixel inputs (e.g., images) problems (He et al., 2020; Chen et al., 2020). In our work, we train policy representation with contrastive learning where data augmentation is performed on policy data. For SPR, i.e., state-action pairs as policy data, there is no need to perform data augmentation since different batches of randomly sampled state-action pairs naturally forms positive samples, since they all reflect the behavior of the same policy. A similar idea in Meta-RL is in (Fu et al., 2020).

For OPR, i.e., policy network parameters as policy data, it is unclear how to perform data augmentation on them. In this work, we consider two kinds of data augmentation for policy network parameters as shown in Figure 10. We found similar results for both random mask and noise corruption, and we use random mask as default data augmentation in our experiments.

As an unsupervised representation learning method, contrastive Learning encourages policies to be close to similar ones (i.e., positive samples π^+) and to be apart from different ones (i.e., negative samples π^-) in policy representation space. The policy representation network is then trained with InfoNCE loss (Oord et al., 2018), i.e., to minimize the cross-entropy loss below:

$$\mathcal{L}_{\text{CL}} = \log \frac{\exp(\chi_\pi^T W \chi_{\pi^+})}{\exp(\chi_\pi^T W \chi_{\pi^+}) + \sum_{\pi^-} \exp(\chi_\pi^T W \chi_{\pi^-})}$$

D.4 Pseudo-code of Policy Representation Learning Framework

The pseudo-code of the overall framework of policy representation learning is in Algorithm 3. Note that in Line 21, the positive samples $\chi_{\pi_i^+}$ is obtained from a momentum policy encoder (He et al.,

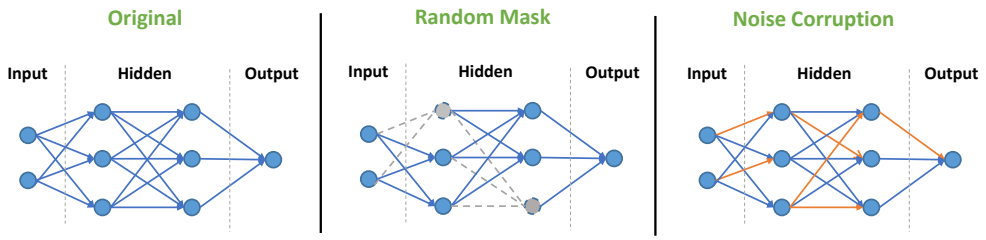


Figure 10: Examples of data augmentation on policy network parameters for Origin Policy Representation (OPR). *Left*: an example of original policy network. *Middle*: dropout-like random masks are performed on original policy network, where gray dashed lines represent the weights masked out. *Right*: randomly selected weights are corrupted by random noises, denoted by orange lines.

2020) with another augmentation for corresponding policy data, while negative samples $\chi_{\pi_i^-}$ are other policy embeddings in the same batch, i.e., $\chi_{\pi_i^-} \in B \backslash \{\chi_{\pi_i}\}$.

Algorithm 3 A Framework of Policy Representation Learning

Input: policy dataset $\mathbb{D} = \{(\pi_i, \omega_i, \mathcal{D}_{\pi_i})\}_{i=1}^n$, consisting of policy π_i, policy parameters ω_i and state-action pairs $\mathcal{D}_{\pi_i} = \{(s_j, a_j)\}_{j=1}^m$

Initialize the policy encoder g_α with parameters α
Initialize the policy decoder (or master policy) (network) $\bar{\pi}_\beta(a|s, \chi_\pi)$ for SPR and the weight matrix W for ORP respectively
for iteration $i = 0, 1, 2, \ldots$ **do**
    Sample a mini-batch of policy data $\mathcal{B}$ from $\mathbb{D}$
    # Encode and obtain the policy embedding χ_{π_i} with SPR or OPR
    if Use OPR **then**
        if Use Contrastive Learning **then**
            Perform data augmentation on each $w_i \in \mathcal{B}$
        end if
        $\chi_{\pi_i} = g_\alpha^{\text{OPR}}(\omega_{\pi_i})$ for each $(\pi_i, \omega_i, \cdot) \in \mathcal{B}$
    else if Use PPR **then**
        $\chi_{\pi_i} = g_\alpha^{\text{SPR}}(B_i)$ where B_i is a mini-batch of state-action pairs sampled from $\mathcal{D}_{\pi_i}$, for each $(\pi_i, \cdot, \mathcal{D}_{\pi_i}) \in \mathcal{B}$
    end if
    # Train policy encoder g_α in different ways (i.e., AUX or CL)
    if Use Auxiliary Loss **then**
        Sample a mini-batch of state-action pairs $B = (s_i, a_i)_{i=1}^b$ from $\mathcal{D}_{\pi_i}$ for each π_i
        Compute the auxiliary loss, $\mathcal{L}_{\text{Aux}} = -\sum_{(s_i,a_i)\in B} \log \bar{\pi}_\alpha(a_i|s_i, \chi_{\pi_i})$
        Update parameters α, β to minimize $\mathcal{L}^{\text{Aux}}$
    end if
    if Use Contrastive Learning **then**
        Calculate contrastive loss, $\mathcal{L}_{\text{CL}} = -\sum_{\chi_{\pi_i}\in B} \log \frac{\exp(\chi_{\pi_i}^T W \chi_{\pi_i^+})}{\exp(\chi_{\pi_i}^T W \chi_{\pi_i^+}) + \sum_{\pi_i^-} \exp(\chi_{\pi_i}^T W \chi_{\pi_i^-})}$, where $\chi_{\pi_i^+}, \chi_{\pi_i^-}$ are positive and negative samples
        Update parameters α, W to minimize $\mathcal{L}^{\text{CL}}$
    end if
    # Train policy encoder g_α with the PeVFA approximation loss (E2E)
    Calculate the value approximation loss of PeVFA, $\mathcal{L}$
    Update parameters α to minimize $\mathcal{L}$
end for

D.5 A REVIEW OF RELATED WORKS ON POLICY REPRESENTATION/EMBEDDING LEARNING

A few works involve representation or embedding learning for RL policy or skill in Fast Adaptation and Transfer (Arnekvist et al., 2019; Raileanu et al., 2020; Hausman et al., 2018), Multiagent Learning (Grover et al., 2018), Hierarchical RL (Wang et al., 2020). We provide a brief review for above works below.

E COMPLETE BACKGROUND

E.1 REINFORCEMENT LEARNING

Markov Decision Process. We consider a Markov Decision Process (MDP) defined as $\langle \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \gamma \rangle$ with $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, r the reward function, $\mathcal{P}$ the transition function and $\gamma \in [0, 1)$ the discount factor. A policy $\pi \in \Pi = P(\mathcal{A})^{|\mathcal{S}|}$, defines the distribution over all actions for each state. An RL agent interacts with the environment with its policy and generates the trajectory $s_0, a_0, r_1, s_1, a_1, r_2, ...$, and seeks for an optimal policy that maximizes the expected long-term discounted return, $\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \right]$, where $r_{t+1} = r(s_t, a_t)$.

Value Function. Almost all RL algorithms involve value functions (Sutton & Barto, 1998), which estimate how good a state or a state-action pair is conditioned on a given policy. The *state-value function* $v^\pi(s)$ is defined in terms of the expected return obtained through following the policy π from a state s:

$$v^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s \right] \text{ for all } s \in \mathcal{S}.$$

Similarly, *action-value function* is defined for all state-action pairs as $q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s, a_0 = a \right]$. Typically, value functions are learned through Monte Carlo (MC) or Temporal Difference (TD) algorithms (Sutton & Barto, 1998).

Bellman equations defines the recursive relationships among value functions. The *Bellman Expectation equation* of $v^\pi(s)$ has a matrix form as below (Sutton & Barto, 1998):

$$V^\pi = r^\pi + \gamma \mathcal{P}^\pi V^\pi = (\mathcal{I} - \gamma \mathcal{P}^\pi)^{-1} r^\pi, \tag{14}$$

where V^π is a $|\mathcal{S}|$-dimensional vector, $\mathcal{P}^\pi$ is the state-to-state transition matrix $\mathcal{P}^\pi(s'|s) = \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{P}(s'|s, a)$ and r^π is the vector of expected rewards $r^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) r(s, a)$. Equation 14 indicates that value function is determined by policy π and environment models (i.e., $\mathcal{P}$ and r. For a conventional value function, all of them are modeled implicitly within a table or a function approximator, i.e., a mapping from only states (and actions) to values.

E.2 AN UNIFIED VIEW OF EXTENSIONS OF CONVENTIONAL VALUE FUNCTION FROM THE VECTOR FORM OF BELLMAN EQUATION

Consider explicit representation of some of them during value estimation can develop the ability of conventional value functions and solve challenging problems, e.g., goal-conditioned RL (Schaul et al., 2015; Andrychowicz et al., 2017), Hierarchical RL (Nachum et al., 2018; Wang et al., 2020), opponent modeling (He & Boyd-Graber, 2016; Grover et al., 2018; Tacchetti et al., 2019), and context-based Meta-RL (Rakelly et al., 2019; Lee et al., 2020).

As discussed in Sec. 2.2, most extensions of conventional VFA are proposed for the purpose of value generalization. Therefore, we provide a unified view of such extensions through looking into the vector form *Bellman equation* of $v^\pi(s)$ (Sutton & Barto, 1998) as Equation 14. We can see that different extensions focus Equation 14. Consider value function is a function of policy π and environment models (i.e., $\mathcal{P}$ and r):

- Goal-conditioned RL and context-based meta-RL focus on value generalization among representations for environment models, i.e., the goal representation and the context.
- Opponent modeling, Ad-hoc team seek to generalize among different opponents or teammates in a Multiagent System, with learned representation of opponents. This can be viewed as a special case of value generalization among environment models sicne from one agent view, other opponents are part of the environment which also determines the dynamics and rewards.

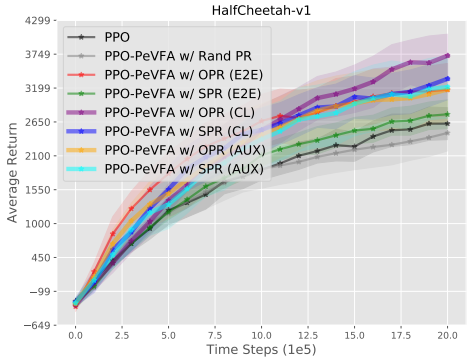
- Hierarchical RL is also a special case of value generalization among environment models. The idea is to view the separate learning process of hierarchical policies for different levels as a multiagent learning system.

The common thing of above is that, they learn a representation of the environment (external variants). In contrast, we study value generalization among own policies (leanring dynamics), which cares about internal invariants.

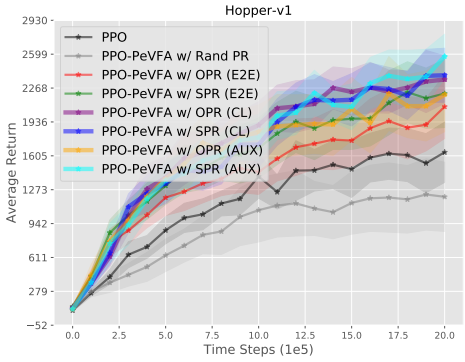
F Experimental Details and Complete Results

F.1 An Overall View of Learning Curves of Different Algorithms

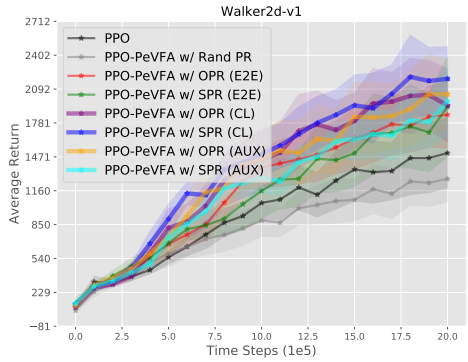
F.2 Visualization of Learned Policy Representation



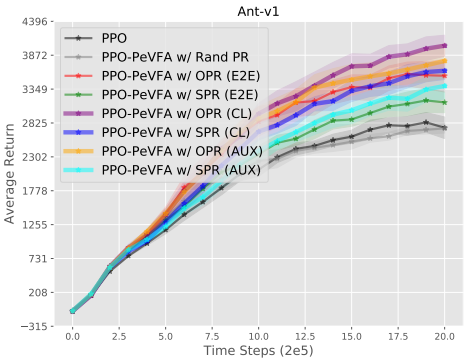
(a) HalfCheetah-v1



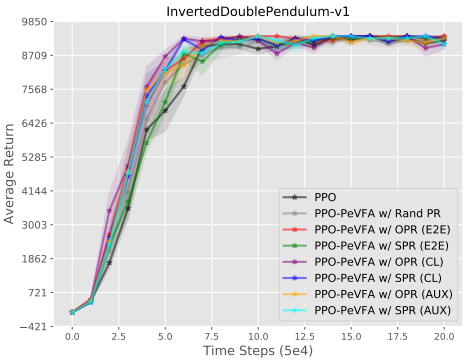
(b) Hopper-v1



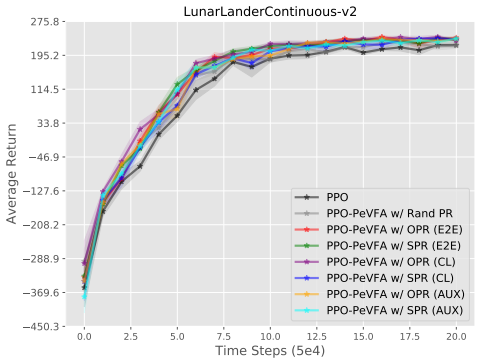
(c) Walker2d-v1



(d) Ant-v1

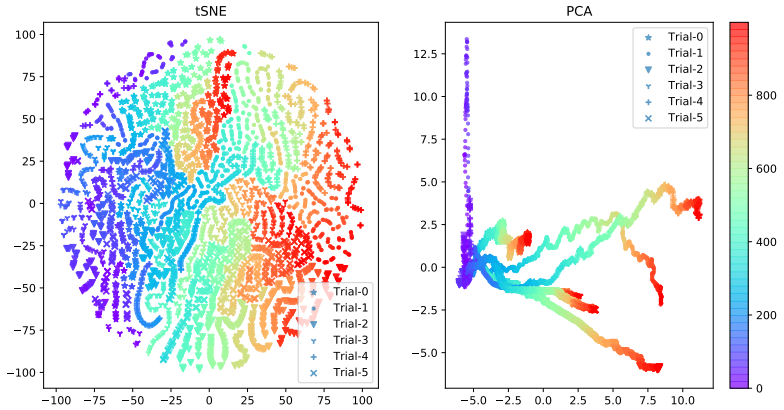


(e) InvertedDoublePendulum-v1

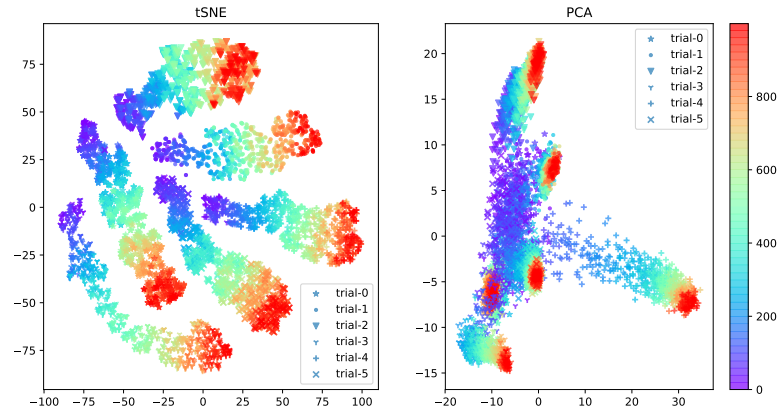


(f) LunarLanderContinuous-v2

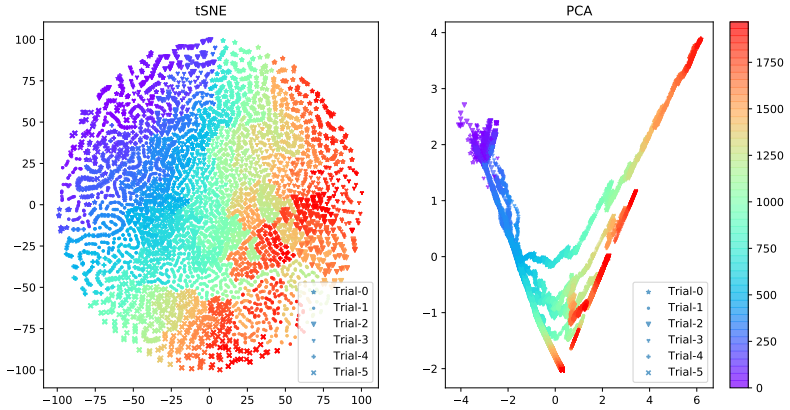
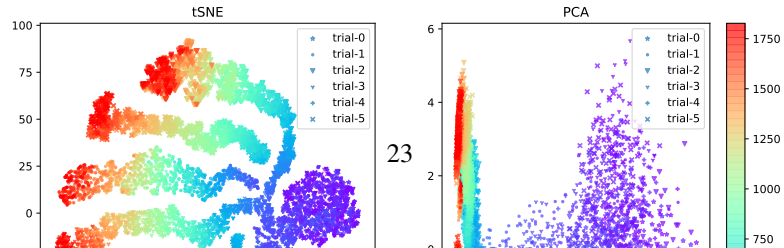
Figure 11: An overall view of performance evaluations of different algorithms in MuJoCo continuous control tasks. The results are average returns and the shaded region denotes half a standard deviation over 10 trials.



(a) OPR (E2E) at HalfCheetah-v1



(b) SPR (E2E) at HalfCheetah-v1



(c) OPR (E2E) at Ant-v1