Boosting Off-policy RL with Policy Representation and Policy-extended Value Function Approximator

Min Zhang [1] **Jianye Hao** [1] **Hongyao Tang** [1] **Yan Zheng** [1]

Abstract

Off-policy Reinforcement Learning (RL) is fundamental to realizing intelligent decision-making agents by trial and error. The most notorious issue of off-policy RL is known as *Deadly Triad*, i.e., Bootstrapping, Function Approximation, and Off-policy Learning. Despite recent advances in bootstrapping algorithms with better bias control, improvements on the latter two factors are relatively less studied. In this paper, we propose a general off-policy RL algorithm based on policy representation and policy-extended value function approximator (PeVFA). Orthogonal to better bootstrapping, our improvement is two-fold. On one hand, PeVFA's nature in fitting the value functions of multiple policies according to corresponding low-dimensional policy representation offers preferable function approximation with less interference and better generalization. On the other hand, PeVFA and policy representation allow to perform off-policy learning in a more general and sufficient manner. Specifically, we perform additional value learning for proximal historical policies along the learning process. This drives the value generalization from learned policies and in turn, leads to more efficient learning. We evaluate our algorithms on continuous control tasks and the empirical results demonstrate consistent improvements in terms of efficiency and stability.

1. Introduction

Off-policy Reinforcement Learning is an important branch of reinforcement learning that has attracted much attention thanks to its generality and application potential (François-Lavet et al., 2018). In off-policy RL, three widely utilized techniques: Bootstrapping, Function Approximation, and Off-policy Learning, are combined together, which is collectively referred to as *Deadly Triad* (Sutton, 1988; Sutton & Barto, 2018). Despite high-profile empirical successes, sample inefficiency and learning instability due to Deadly Triad remain to be key issues of off-policy RL (Achiam et al., 2019; Van Hasselt et al., 2018). This greatly limits the deployment of off-policy RL in real-world scenarios. In recent years, significant progress has been made by improving bootstrapping methods, resulting in more advanced off-policy RL algorithms (Fujimoto et al., 2018; Kuznetsov et al., 2020; Lan et al., 2020; Chen et al., 2021; Liang et al., 2022). Nevertheless, the latter two factors (i.e., function approximation and off-policy learning) receive relatively less attention. Existing works on these two factors primarily focus on how to design better function approximators (Ota et al., 2020; Shah & Kumar, 2021) (usually modeled by neural networks) and compensate for the discrepancy between the distributions of the policy of interest and the behavior policy (Saglam et al., 2022; Kumar et al., 2020). Although some progress has been made in these works, essentially they have only independently improved one of the two factors. We argue that the two factors are closely related and that improving both of them simultaneously may obtain greater improvements.

Recently, in the on-policy setting, Faccio et al. (2020) and Tang et al. (2020) both propose methods to improve function approximation by using policy parameters and policy representations with encoded policy parameters as an additional input to the traditional Value Function Approximator (VFA). Similarly, in the offline setting, the Policy Evaluation Networks (PVN) (Harb et al., 2020) is proposed to approximate the expected return of multiple policies with policy representation obtained by policy fingerprints as input. Furthermore, towards obtaining preferable function approximation with better generalization across policies and tasks, Raileanu et al. (2020) and Sang et al. (2022) propose to use both a policy representation and an environment representation as additional inputs to the VFA. On the surface, these works mentioned above develop different value function approximators, but essentially they aim to improve value approximation by extending the input of the VFA with policy and environment representations.

[1]College of Intelligence and Computing, Tianjin University, Tianjin, China. Correspondence to: Jianye Hao <jianye.hao@tju.edu.cn>.

Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning (ICML), Honolulu, Hawaii, USA, 2023. Copyright 2023 by the author(s).

Inspired by these works, we investigate how the policy-extended value function approximator (PeVFA) can improve the off-policy RL from both function approximation and off-policy learning. First of all, we propose a new Bellman operator to characterize off-policy RL with PeVFA for the purpose of better function approximation. Building upon the proposed Bellman operator, we further develop a **G**eneralized **O**ff-**P**olicy **E**valuation manner with **P**eVFA (**GOPE-P**) for more efficient learning. Additionally, we propose a simple and effective policy representation learning method, named **L**ayer-wise **P**ermutation-invariant **E**ncoder with **D**ynamic **M**asking (**LPE-DM**), which follows the characteristics of policy data itself for learning policy representation. Our work differs from previous studies in two aspects. Firstly, we focus on the more general off-policy scenario. Secondly, in addition to function approximation, we further investigate the effectiveness of PeVFA in off-policy learning. We refer to our method as **Off-policy PeVFA**.

In this paper, we first propose to improve both function approximation and off-policy learning from the PeVFA perspective. To evaluate the effectiveness and generality of the proposed method, we introduce two practical implementations of our method based on TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018), namely TD3-PeVFA and SAC-PeVFA. We evaluate them on six OpenAI continuous control tasks and the empirical results indicate that our algorithm significantly outperforms the benchmarks in every environment tested. We summarize our main contributions below:

1) We propose a general off-policy RL algorithm based on PeVFA, which leverages value generalization among policies to improve the learning process.

2) We propose a new policy representation learning method for the effective encoding of policy network parameters.

3) Empirical results on popular OpenAI Gym control tasks demonstrate the consistent superiority of our algorithms in terms of efficiency and stability.

2. Preliminary

2.1. Reinforcement Learning

A Markov Decision Process (MDP) (Puterman, 2014) is usually 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 $\pi\in\Pi: S\to\Delta(A)$ is a mapping from states to action distributions, which defines how to behave 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 $r_{t+1}=r(s_t,a_t)$, where ρ_0 is the initial state distribution. The goal of an RL agent is to maximize the value defined as the expected cumulative discounted returns $\mathbb{E}_{s_0\sim\rho_0(\cdot)}[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\mid s_0]$. Given a policy π, the discounted state visitation distribution from initial states regarding ρ_0 is defined as $d^\pi(s)$. There exists an action-value function $Q^\pi(s,a)=\mathbb{E}_{s\sim d^\pi(s),a\sim\pi}[\sum_{t=0}^{\infty}\gamma^t r_{t+1}\mid s_0=s,a_0=a]$, which is defined as the expected return for performing action a in state s. We compute the action-value function through the Bellman operator, $\mathcal{T}^\pi Q^\pi(s,a)=\mathbb{E}_{s'\in S}[r+\gamma Q^\pi(s',\pi(s'))]$ (Sutton & Barto, 2018). The optimal action-value function $Q^*(s,a)=\max_{a\in A}Q^\pi(s,a)$ is obtained by the greedy actions of the corresponding policy.

2.2. Off-policy RL

In deep RL, the action-value function is modeled by a differentiable function approximator $Q_\theta(s,a)$ with parameters θ, commonly known as Q-network. The deep action-value function is obtained by temporal difference (TD) learning (Sutton, 1988) based on the Bellman equation:

$$Q_\theta(s,a)\leftarrow(r+\gamma Q_{\theta^-}(s',a')),\forall s,s'\in\mathcal{S},a,a'\in\mathcal{A}, \quad (1)$$

where Q_{θ^-} is the target network for providing a fixed objective to the Q-network and ensuring stability in the updates. In particular, the target network is updated by some proportion τ at each time step $\theta^-\leftarrow\tau\theta+(1-\tau)\theta^-$, named soft-update. The Q-network is optimized by minimizing *TD-error*, i.e., $(r+\gamma Q_{\theta^-}(s',a'))-Q_\theta(s,a)$.

In continuous action spaces, obtaining the maximum of the action-value function over the action space, i.e., $\max_{a\in A}Q^\pi(s,a)$, is intractable (Saglam et al., 2022). Thus, a separate network named the actor network is employed which selects actions on the observed states. The deep RL algorithms that adopt both actor and critic are called actor-critic methods. In actor-critic methods, the actor network π_ϕ with parameters ϕ is optimized by one-step gradient ascent over the policy gradient $\nabla_\phi J(\phi)$, where $J(\phi)=Q_\theta(s,\pi_\phi(s))$. For deterministic policies, i.e., mapping states to unique actions, the policy gradient following the Deterministic Policy Gradient algorithm (DPG) (Silver et al., 2014) is computed: $\nabla_\phi J_{det}(\phi)=\nabla_a Q_\theta(s,a)|_{a=\pi_\phi(s)}\nabla_\phi\pi_\phi(s)$. For stochastic policies, i.e., mapping states to action probabilities, the policy gradient is computed: $\nabla_\phi J_{sto}(\phi)=Q_\theta(s,a)\nabla_\phi\log\pi_\phi(a\mid s)$.

In off-policy deep RL, the policy π_ϕ can be optimized using collected data that are not necessarily obtained under the current policy π_ϕ, but from a *behavioral* policy π_β. In this case, the deterministic policy gradient and stochastic policy gradient are respectively:

$$\nabla_\phi J_{det}(\phi)=\mathbb{E}_{s\sim d^{\pi_\beta}(s)}\left[\nabla_a Q_\theta(s,a)|_{a=\pi_\phi(s)}\nabla_\phi\pi_\phi(s)\right], \quad (2)$$

$$\nabla_\phi J_{sto}(\phi)=\mathbb{E}_{s\sim d^{\pi_\beta}(s),a\sim\pi_\phi}\left[Q_\theta(s,a)\nabla_\phi\log\pi_\phi(a\mid s)\right]. \quad (3)$$

The sample efficiency of off-policy RL algorithms is improved as they make use of any past experience (Degris et al., 2012). This may be especially useful in some scenarios where it may be costly or dangerous to collect data using the learned policy.

2.3. Policy-Extended Value Function Approximator

The policy-extended value function approximator (PeVFA, $\mathbb{V} : S \times \Pi \to \mathbb{R}$) (Tang et al., 2020) is an extension of the conventional value function approximator (VFA), which explicitly takes policy (representation) as input. Here, we consider a general policy representation function $f : \Pi \to \mathcal{X} \in \mathbb{R}^n$ which maps the original policy $\pi \in \Pi$ to a n-dimensional policy representation $\chi_\pi \in \mathcal{X}$. Following the definition of PeVFA (Tang et al., 2020), for $\forall s \in S, a \in A, \pi \in \Pi$, we define the policy-extended action-value function $\mathbb{Q}(s, a, \chi_\pi)$:

$$\mathbb{Q}(s, a, \chi_\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} \mid s_0 = s, a_0 = a \right]. \quad (4)$$

One key difference from the original action-value function $Q(s, a)$ is that $\mathbb{Q}(s, a, \cdot)$ preserves the values of multiple policies. With function approximation, $\mathbb{Q}_\theta(s, a, \cdot)$ is expected to approximate values among policy space:

$$F_{\mu,p,\rho}(\theta, f, \Pi) = \sum_{\pi \in \Pi} \mu(\pi) \left\| \mathbb{Q}_\theta(s, a, \chi_\pi) - Q^\pi(s, a) \right\|_{p,\rho}, \quad (5)$$

$$\theta^* = \arg\min_{\theta \in \Theta} F(\theta, f, \Pi), \quad (6)$$

where μ, ρ are distributions over policies and states respectively, which are related to specific tasks. p denotes L_p-norm. From Eq.6, policy representations endow $\mathbb{Q}_\theta(s, a, \cdot)$ with the property of value generalizing across policies, allowing $\mathbb{Q}_\theta$ to utilize preserved old knowledge within the value function to evaluate unseen policies efficiently. This appealing characteristic is highly significant for improving Generalized Policy Iteration (GPI) (Sutton, 1988) followed by most RL algorithms, because the value function $\mathbb{Q}_\theta$ can quickly generalize to successive policies by leveraging the value of previous policies along the policy improvement path during GPI. Thus, in this paper, we focus on the GPI and use a uniform state distribution ρ and L_2-norm. Intuitively, for $\mu(\pi)$, it should be the distribution of policies we are interested in when learning the value function. We defer the concrete choices of it in Sec. 3.2. For brevity, we refer to the policy-extended action-value function as PeVFA in the following.

3. Off-Policy RL with PeVFA

Our goal is to propose a general off-policy RL algorithm with policy representations and PeVFA for boosting off-policy RL. In this section, we first formally define the new Bellman operator with PeVFA, $\mathbb{T}^\pi$ which is used to compute the action-value function in off-policy RL (Sec. 3.1). Then, we propose a more general and efficient manner to perform off-policy learning based on the PeVFA (Sec. 3.2). Finally, we present in detail a practical implementation of the proposed algorithm (Sec. 3.3).

3.1. Bellman Operator with PeVFA

In general, off-policy RL algorithms learn the conventional action-value function Q^π through the Bellman operator $\mathcal{T}^\pi$: $\mathcal{T}^\pi Q^\pi(s, a) = \mathbb{E}_{s' \in S}\left[r + \gamma Q^\pi(s', \pi(s'))\right]$. Thus, we introduce our algorithm starting from the Bellman operator. First of all, following the definition of the Bellman operator, we propose a novel Bellman operator $\mathbb{T}^\pi$ regarding the policy-extended action-value function as follows:

Definition 3.1 (Bellman operator with PeVFA)**.** Let $\mathbb{T}^\pi : \mathbb{Q} \to \mathbb{Q}$ be the operator on the policy-extended action-value function $\mathbb{Q}$. For a given policy $\pi \in \Pi$, $f(\pi)$ is a policy representation function that maps the policy π to a n-dimensional policy representation $\chi_\pi \in \mathcal{X}$. $\forall s, s' \in S, a, a' \in A$, the new Bellman operator $\mathbb{T}^\pi$ is defined as:

$$\mathbb{T}^\pi \mathbb{Q}^\pi(s, a, \chi_\pi) = \mathbb{E}_{s' \in S}\left[r + \gamma \mathbb{Q}^\pi(s', \pi(s'), \chi_\pi)\right]. \quad (7)$$

$\mathbb{T}^\pi$ is a recursive operator which satisfies the *compression map theorem* (Sutton & Barto, 2018) with a unique fixed-point $\mathbb{Q}^\pi$, denoted as $\mathbb{T}^\pi \mathbb{Q}^\pi = \mathbb{Q}^\pi$. Hence, for arbitrary policy π, we perform multiple Bellman operations on its initial policy-extended action-value function $\mathbb{Q}_0^\pi$ to obtain the unique fixed-point $\mathbb{Q}^\pi$, i.e., convergence to the policy-extended action-value function of the policy π.

The learning process of action-value function Q^π based on the two Bellman operators $\mathcal{T}^\pi$ and $\mathbb{T}^\pi$ can be expressed as $\lim_{n\to\infty} \mathcal{T}_n^\pi Q_0^\pi = Q^\pi$ and $\lim_{n\to\infty} \mathbb{T}_n^\pi \mathbb{Q}_0^\pi = Q^\pi$, respectively. Somewhat naturally, the closer $\mathcal{Q}_0^\pi$ and $\mathbb{Q}_0^\pi$ are to the true value Q^π, the smaller n is, resulting in the higher efficiency of value iteration. Thus, a key question is whether $\mathbb{Q}_0^\pi$ is closer to the true value Q^π than $\mathcal{Q}_0^\pi$. We first study the value learning in a two-policy case (i.e., π_t,π_{t+1}) where only the value of policy π_1 is approximated by PeVFA as below:

Theorem 3.2. *For a value learning process, $\mathcal{P}_\pi : \Theta \to \hat{\Theta}, f_{\hat{\theta}}(\pi) = \|\mathbb{Q}_{\hat{\theta}}(\pi) - Q^\pi\|$, if $f_{\hat{\theta}_t}(\pi_t) + f_{\hat{\theta}_t}(\pi_{t+1}) \leq \|Q^{\pi_t} - Q^{\pi_{t+1}}\|$, then $\|\mathbb{Q}_{\hat{\theta}_t}(\pi_{t+1}) - Q^{\pi_{t+1}}\| \leq \|\mathbb{Q}_{\hat{\theta}_t}(\pi_t) - Q^{\pi_{t+1}}\|$.*

Theorem 3.2 can be proved by the *Triangle Inequality*. Based on the Theorem 3.2, the value learning at policy π_{t+1} with PeVFA starts from the generalized value estimation $\mathbb{Q}_{\hat{\theta}_t}(\pi_{t+1})$. Conversely, with respect to VFA, the value learning at policy π_{t+1} starts from the $Q_{\hat{\theta}_t}$(i.e., equivalent to $\mathbb{Q}_{\hat{\theta}_t}(\pi_t)$) which has no explicit policy π_{t+1} as an additional input. Thus, PeVFA offers a better initial function approximation. Recall the GPI process, the value learning from each π_t and π_{t+1} can be similarly considered as the two-policy case.

3.2. Generalized Off-policy Evaluation with PeVFA

In this section, we mainly investigate *how to make use of PeVFA to improve off-policy learning.* To the best of our knowledge, there is no related work to improve off-policy learning from the PeVFA perspective. Recall the PeVFA introduced in Sec.2.3, it is characterized by fitting the value functions of multiple policies based on corresponding policy representation χ_π, i.e., $\theta^* = \arg\min_{\theta\in\Theta} \sum_{\pi\in\Pi} \mu(\pi) \left\|\mathbb{Q}_\theta\left(s,a,\chi_\pi\right) - Q^\pi(s,a)\right\|_{p,\rho}$. Naturally, we argue that performing value learning across policies can motivate PeVFA to offer more perfect function approximation. Inspired by this, we propose a generalized off-policy evaluation manner with PeVFA, denoted as **GOPE-P**.

Concretely, during the GPI ($\theta_0 \xrightarrow{\mathbb{T}_n^{\pi_1}} \theta_1, \cdots, \theta_t \xrightarrow{\mathbb{T}_n^{\pi_t}} \theta_{t+1}, \cdots$), for a value approximation process at each iteration $\theta_t \xrightarrow{\mathbb{T}_n^{\pi_t}} \theta_{t+1}$, we add an additional value learning process $\mathcal{P}_{\pi\in\Pi_{S_t}^{GPI}}$, i.e., $\theta_t \xrightarrow{\mathcal{P}_{\pi\in\Pi_{S_t}^{GPI}}} \theta_t' \xrightarrow{\mathbb{T}_n^{\pi_t}} \theta_{t+1}$. $\Pi_{S_t}^{GPI}$ denotes the policy subspace obtained along the GPI at the t-th iteration. Similarly, $\forall\pi_i \in \Pi_{S_t}^{GPI}$, we update the PeVFA for $\mathcal{P}_{\pi\in\Pi_{S_t}^{GPI}}$ using Bellman operator with PeVFA:

$$\mathbb{Q}_\theta^{\pi_i}(s,a,\chi_{\pi_i}) \leftarrow r+\gamma\mathbb{Q}_{\theta^-}^{\pi_i}\left(s',\pi_i(s'),\chi_{\pi_i}\right), \forall(s,a,r,s')\in\mathcal{B}, \tag{8}$$

where $\mathcal{B}$ indicates the experience replay buffer shared by all policies. With PeVFA, we extend off-policy learning to any policy in the policy subspace derived from GPI. With the proposed **GOPE-P**, our method further improves learning efficiency while improving the generalization of the value function across policies. We defer the discussion of the policy subspace in Appendix C.

3.3. A Practical Implementation of the Proposed Algorithm

The Sec.3.1 and 3.2 detail how PeVFA boosts the off-policy RL from both function approximation and off-policy learning. Next, combining the general and popular value estimation method, Clipped Double Q-learning (CDQ) adopted in off-policy RL, such as the TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018), we propose a practical implementation of the proposed algorithm.

We first recall the CDQ method. Following Double Q-learning algorithm (Hasselt, 2010; Van Hasselt et al., 2016), the CDQ consists of double estimator, i.e., $Q_{\theta_1}, Q_{\theta_2}$. However, instead of using the opposite critic in the learning targets, which suffer from overestimation, the CDQ proposes to simply upper-bound the less biased value estimate Q_{θ_2} by the biased estimate Q_{θ_1} to alleviate the overestimation problem. Thus, for any policy π_ϕ during the learning

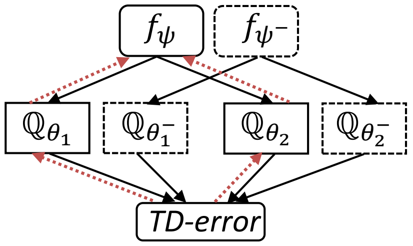

Figure 1. The illustration diagram for the PeVFA-based Clipped Double Q-learning. The black solid arrows indicate forward propagation and red dashed arrows are backward propagation.

process, the update target that both critic $Q_{\theta_1}, Q_{\theta_2}$ share is:

$$y = r + \gamma \min_{i=1,2} Q_{\theta_i^-}\left(s', \pi_\phi\left(s'\right)\right). \tag{9}$$

In this work, replacing the VFA (i.e., $Q_{\theta_i}, Q_{\theta_i^-}$) with PeVFA (i.e., $\mathbb{Q}_{\theta_i}, \mathbb{Q}_{\theta_i^-}$), we propose a PeVFA-based CDQ method, named **PeCDQ**. Furthermore, the value approximation of both critics is formulated below, for i=1, 2:

$$\mathbb{Q}_{\theta_i}(s,a,\chi_\pi) \leftarrow r + \gamma \min_{i=1,2} \mathbb{Q}_{\theta_i^-}\left(s', \pi_\phi\left(s'\right), \chi_\pi\right). \tag{10}$$

For policy representation χ_π, in this paper, we propose to learn policy representation from policy parameters ϕ, i.e., $\chi_\pi = f_\psi(\phi)$. The $f(\cdot)$ denotes the policy encoder with parameter ψ. In particular, in order to adapt to the target PeVFA network $\mathbb{Q}_{\theta_i^-}$ in the PeCDQ, we maintain a target policy encoder f_{ψ^-}. This makes our modified TD learning update:

$$\mathbb{Q}_{\theta_i}(s,a,f_\psi(\phi)) \leftarrow r+\gamma \min_{i=1,2} \mathbb{Q}_{\theta_i^-}\left(s', \pi_\phi\left(s'\right), f_{\psi^-}(\phi)\right). \tag{11}$$

The target policy encoder $f_{\psi^-}(\phi)$ is updated by $\psi^- \leftarrow \tau\psi + (1-\tau)\psi^-$. Fig.1 illustrates the structure of PeCDQ. Additionally, we also empirically investigate other structures and perform experimental validation on four Mujoco-based environments. More method details and experimental results are in the Appendix B.

To demonstrate the generality of our algorithm, we implement **TD3-PeVFA** and **SAC-PeVFA** in this paper. Due to space limitations, we present only **TD3-PeVFA** in the Algorithm 3.3, and **SAC-PeVFA** can be found in Appendix F.

4. Dynamic Masking for Policy Network Representation Learning

To derive general off-policy deep RL algorithms with the proposed **GOPE-P**, a tricky question is how to learn compact policy representations for better approximation and generalization of PeVFA across policies. Intuitively, policy representations optimized based on TD learning contain the most relevant features with value approximation. Therefore, we consider using TD loss to optimize policy

Algorithm 1 TD3-PeVFA

```
Initialize critic networks $\mathbb{Q}_{\theta_1}$,$\mathbb{Q}_{\theta_2}$, actor network $\pi_\phi$ and policy encoder network $f_\psi$, with random parameters $\theta_1$, $\theta_2$, $\phi$, $\psi$, target networks ${\theta_1}^- \leftarrow \theta_1$, ${\theta_2}^- \leftarrow \theta_2$, $\phi^- \leftarrow \phi$, $\psi^- \leftarrow \psi$, replay buffer $\mathcal{B}$, policy buffer $\mathcal{D}$
for iteration $t = 0, 1, 2, \cdots$ do
  Select action $a$ and observe reward $r$ and new state $s'$. Store transition tuple $(s, a, r, s')$ in $\mathcal{B}$. Store policy $(\phi, \phi^-)$ in $\mathcal{D}$
  Value learning of historical policies
  for $m = 0, 1, 2, \cdots$ do
    Sample mini-batch of $n$ policies from $\mathcal{D}$ and mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
    for $v = 1, 2, \cdots, n$ do
      $a' \leftarrow \pi_{\phi_v^-}(s')$, store $N$ transition tuple $(s, a, r, s', a', v)$ in $\mathcal{B}'$
    end for
    Sample mini-batch of $N$ transitions $(s, a, r, s', a', v)$ from $\mathcal{B}'$
    $y \leftarrow r + \gamma \min_{i=1,2} \mathbb{Q}_{\theta_i^-}\left(s', a', f_{\psi^-}(\phi_v)\right)$
    Update critics $\theta_i, \psi \leftarrow \mathrm{argmin}_{\theta_i,\psi} \mathbb{E}_{(s,a,r,s',a',v)\sim\mathcal{B}'} \sum_{i=1}^{i=2} \left(y - \mathbb{Q}_{\theta_i}\left(s, a, f_\psi(\phi_v)\right)\right)^2$
    if $t$ mod $d$ then
      Update target networks $\theta_i^- \leftarrow \tau\theta_i + (1-\tau)\theta_i^-$, $\psi^- \leftarrow \tau\psi + (1-\tau)\psi^-$
  end for
  Value learning of current policy
  for $m = 0, 1, 2, \cdots$ do
    Sample mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
    $y \leftarrow r + \gamma \min_{i=1,2} \mathbb{Q}_{\theta_i^-}\left(s', \pi_{\phi^-}(s'), f_{\psi^-}(\phi)\right)$
    Update critics $\theta_i, \psi \leftarrow \mathrm{argmin}_{\theta_i,\psi} \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}} \sum_{i=1}^{i=2} \left(y - \mathbb{Q}_{\theta_i}\left(s, a, f_\psi(\phi)\right)\right)^2$
    if $t$ mod $d$ then
      Update actor $\phi \leftarrow \mathrm{argmax}_\phi \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}(\mathbb{Q}_{\theta_1}\left(s, \pi_\phi(s), f_\psi(\phi)\right))$
      Update target networks $\theta_i^- \leftarrow \tau\theta_i + (1-\tau)\theta_i^-$, $\phi^- \leftarrow \tau\phi + (1-\tau)\phi^-$, $\psi^- \leftarrow \tau\psi + (1-\tau)\psi^-$
  end for
end for
```

representations. To be specific, given a policy π_ϕ, the policy representation optimization objective $J(\psi)$ is defined as:

$$J(\psi) = \mathbb{E}_{(s,a,r,s')\in\mathcal{B}}[\sum_{i=1}^{2}(y - \mathbb{Q}_{\theta_i}(s, a, f_\psi(\pi_\phi)))^2] \quad (12)$$

$$y = r + \gamma \min_{i=1,2} \mathbb{Q}_{\theta_i^-}\left(s', \pi_\phi(s'), f_{\psi^-}(\pi_\phi)\right). \quad (13)$$

For practical implementation, we further consider 1) the policy data sources for learning policy representations and 2) the construction of f_ψ. In essence, any data characterizing policies can be used as the policy data sources, such as parameters of a policy network, trajectories, etc. However, compared with the trajectories with high randomness, the parameters of the policy network are usually available and highly deterministic, which is more convenient for learning policy representation. Typically, in DRL, the size of policy networks is usually 64×64, 256×256, or larger. The high-dimensional policy parameters raise the question of whether to always use all parameters to learn policy representations during the learning process. This motivates a hypothesis:

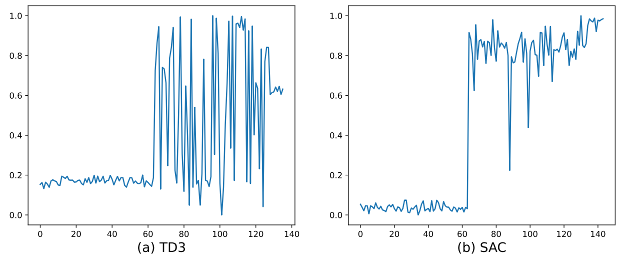

Figure 2. Average amount of parameter change of policy network neural nodes on the Ant during the learning process. The x-axis represents the indices of neural nodes, and the y-axis is the normalized average amount of parameter change.

some neural nodes of the policy network may not be active in parameter updates and action decisions during the learning process, and the policy parameters associated with these neural nodes may not be useful to the learning policy representations.

To verify the hypothesis, we investigate the evolvement of neural nodes of the policy network during the learning process. We first train a TD3 and a SAC agent on Mujoco-based tasks and store policies $\{\phi_i\}_{i=1,\cdots,N}$ at intervals of

200 steps over the course of training. Ranging from the initial policy π_{ϕ_1} to the final policy π_{ϕ_N}, we iteratively accumulate the amount of parameter change of two adjacent policies, i.e., π_{ϕ_i},$\pi_{\phi_{i+1}}$ for each parameter dimension k, denoted as δ^k. Assuming that the dimensionality of the policy parameters is K, then δ^k is defined as:

$$\delta^k = \sum_{i=1}^{i=N-1} |\phi_i^k - \phi_{i+1}^k|, k = 1 \cdots K, \tag{14}$$

where ϕ_i^k represents the parameter of the kth dimension for the policy π_{ϕ_i}. Further, we calculate the average amount of parameter change in the association parameters for each neural node j $(j = 1, \ldots, J)$ of the policy network, denoted as $\bar{\delta}_j$. In Fig. 2, we show the activity of neural nodes for the policy network on the Ant for both algorithms. The significant differences in the activity of neural nodes of the policy network, illustrated in Fig.2, provide empirical validation of our hypothesis.

Based on the empirical discovery of the activity of neural nodes of the policy network, a natural idea of learning policy representations is to only make use of the neural nodes that are active in parameter change. To this end, we propose *Dynamic Masking (DM)*: a methodology that allows us to dynamically mask neural nodes with low activity during the learning process. The intuitive idea is to characterize the policy using the policy parameters $\widetilde{\phi_i}$ associated with neural nodes with high activity. Formally, $\widetilde{\phi_i}$ is defined as,

$$\widetilde{\phi_i} = \{p_i^j\}, j \notin \{1, 2, \cdots, j, \cdots J\}_{\bar{\delta}_j}^{\eta}, \tag{15}$$

where $\{1, 2, \cdots, j, \cdots J\}_{\bar{\delta}_j}^{\eta}$ denotes node set that are masked by masking ratio η based on the amount of parameter change of nodes $\bar{\delta}_j$. p_i^j denotes the parameters of policy π_{ϕ_i} for associating the jth neural node. As the policy updates, we dynamically update the node set at an interval of 50 steps and the policy representation is always calculated with the latest node set. For a given policy π_ϕ with parameters ϕ, the policy representation can be obtained by $f_\psi(\widetilde{\phi})$. In this work, we adopt a Layer-wise Permutation-invariant Encoder (LPE) (Tang et al., 2020) as the implementation choice of f_ψ. The effectiveness of LPE in encoding policy networks has been demonstrated previously. The implementation details of LPE with DM are presented in Appendix D.

5. Experiments

To evaluate our proposed algorithm, we conduct experiments on the OpenAI Gym continuous control tasks (Brockman et al., 2016). We run each task for 1 million time steps with evaluations every 5000 time steps, where each evaluation reports the average reward over 10 episodes with no exploration noise.

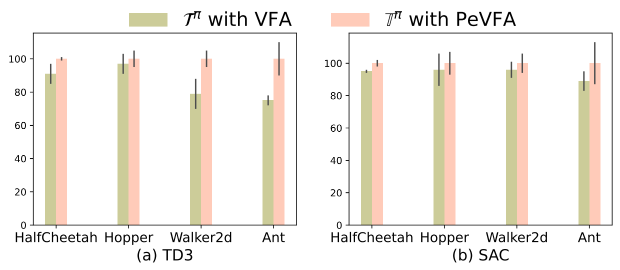

Figure 3. The x-axis represents four tasks, and the y-axis is the normalized average return. Our method ($\mathbb{T}^\pi$ with PeVFA) in each environment is chosen as a normalized baseline. **Conclusion:** In both TD3 and SAC cases, our method outperforms the original Bellman operator ($\mathcal{T}^\pi$ with VFA), which empirically proves that the generalization property of PeVFA across policies.

5.1. Comparative Evaluation

In Sec.3.3, we present a practical implementation of our algorithm, which builds on the representative value estimation algorithm, CDQ (Fujimoto et al., 2018). To be specific, following the original CDQ, we also maintain a pair of critics i.e., $\mathbb{Q}_{\theta_1}(s, a, f_\psi(\pi_\phi))$, $\mathbb{Q}_{\theta_2}(s, a, f_\psi(\pi_\phi))$ along with a single actor π_ϕ. For our implementation of the critic, we utilize a two-layer feedforward neural network of 256 and 256 hidden nodes, respectively. We optimize a 2-layer policy network with 64 hidden nodes for each layer, resulting in over 4k to 10k policy parameters depending on the tasks. In addition, in this paper, for each policy update step, we store the updated policy and maintain a proximal policy buffer D of size 20000. The above setting applies to all experiments in the paper. We list common hyperparameters in Table 5 and Table 6.

The experimental results of our implementations (**TD3-PeVFA**, **SAC-PeVFA**) and the corresponding baselines (**RA** (Random Agent), **TD3** and **SAC**) are reported in Table 1 and Table 2, respectively. We defer the full learning curves to Appendix G (see Fig. 8, 9). In Table 1 and Table 2, we report two performance metrics: 1) the average performance attained over the course of training (denoted **Ave-Evaluation**), which is a measure of the stability of RL algorithms over the course of training and 2) the max performance attained by the algorithm after a fixed number of gradient steps (denoted **Max-Evaluation**). In addition, we evaluate the aggregated improvement (denoted **Norm. Agg.**) of our algorithm on multiple tasks using random agent and TD3, SAC for normalization, respectively. The empirical results demonstrate that our methods (**TD3-PeVFA**, **SAC-PeVFA**) consistently improve the benchmarks on all tasks tested. Moreover, compared with TD3, our method improves 26% and 16% with respect to **Ave-Evaluation** and **Max-Evaluation**, respectively; Compared with SAC, our method improves 23% and 11% with respect to **Ave-Evaluation** and **Max-Evaluation**, respectively. The significant improvement demonstrates the advantages of our algorithm in terms of learning efficiency and stability.

Table 1. Evaluation of TD3-PeVFA in terms of learning stability and efficiency. The results of evaluation returns (± half a std) over 10 trials for algorithms are reported. The best results are bolded for each environment.

Environment	Ave-Evaluation			Max-Evaluation		
	RanP	*TD3*	*TD3-PeVFA*	*RanP*	*TD3*	*TD3-PeVFA*
HalfCheetah	-363.75±83.99	7866.79±546.25	**9048.27±168.56**	-340.60±92.09	9920.17±750.17	**11254.0±159.42**
Hopper	21.43±8.23	2464.53±159.18	**2778.23±101.82**	22.72±8.73	3659.1±28.97	**3666.16±26.83**
Walker2d	-7.76±1.67	2573.99±286.84	**3241.44±194.73**	-6.99± 2.23	4187.83±287.79	**4819.58±156.07**
Ant	926.89±14.53	2265.99±97.2	**3606.0±281.72**	937.77±15.32	3474.56±219.35	**5203.67±365.05**
InvDouPend	72.53±5.79	8463.32±90.4	**8761.04±48.12**	78.34±9.45	9336.3±7.95	**9355.73±1.39**
LunarLander	-206.35±68.91	226.08±9.61	**241.40±6.03**	146.22±41.30	292.26±1.91	**292.58±2.57**
Norm. Agg.	0	1	**1.26** (↑ 26%)	0	1	**1.16** (↑ 16%)

Table 2. Evaluation of SAC-PeVFA in terms of learning stability and efficiency. The results of evaluation returns (± half a std) over 10 trials for algorithms are reported. The best results are bolded for each environment.

Environment	Ave-Evaluation			Max-Evaluation		
	RanP	*SAC*	*SAC-PeVFA*	*RanP*	*SAC*	*SAC-PeVFA*
HalfCheetah	-363.75±83.99	8765.53±133.77	**9908.77±101.9**	-340.60±92.09	12431.71±52.78	**13394.42±476.68**
Hopper	21.43±8.23	2064.33±216.12	**2180.57±98.04**	22.72±8.73	**3503.37±89.04**	3295.36±117.82
Walker2d	-7.76±1.67	3312.06±150.46	**3748.67±189.07**	-6.99± 2.23	5102.93±213.06	**5427.39±217.68**
Ant	926.89±14.53	2416.79±169.40	**3825.16±139.74**	937.77±15.32	3911.47±525.99	**5633.17±174.72**
InvDouPend	72.53±5.79	8983.37±30.96	**9019.36±31.56**	78.34±9.45	9359.37±0.22	**9359.53±0.26**
LunarLander	-206.35±68.91	182.97±13.08	**232.61±11.14**	146.22±41.30	**284.0±1.35**	283.23±1.75
Norm. Agg.	0	1	**1.23** (↑ 23%)	0	1	**1.11** (↑ 11%)

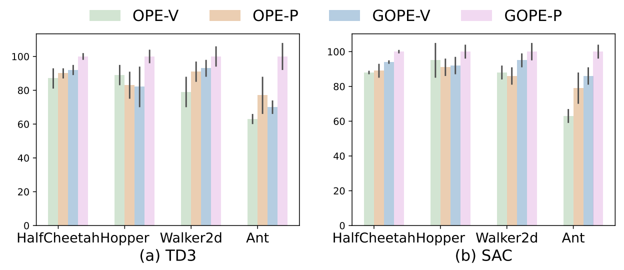

Figure 4. The x-axis represents four test tasks, and the y-axis is the normalized average return. Our method (GOPE-P) in each environment is chosen as a normalized baseline. **Conclusion:** In both TD3 and SAC cases, our method is better than the comparison methods (OPE-V, OPE-P, GOPE-V), which empirically proves the efficacy of generalized off-policy evaluation with PeVFA.

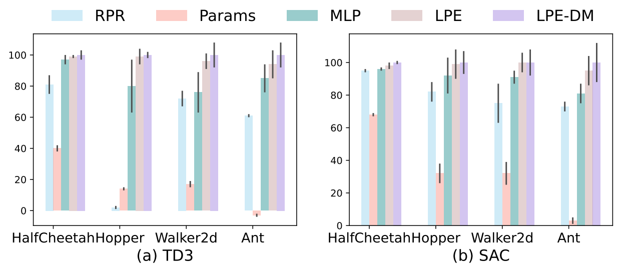

Figure 5. The x-axis represents four test tasks, and the y-axis is the normalized average return. Our method (LPE-DM) in each task is chosen as a normalized baseline. **Conclusion:** In both TD3 and SAC cases, our method is better than the comparison methods (RPR, Params, MLP, LPE), which empirically illustrates that LPE with DM is an efficacy method for learning policy representations from policy parameters.

5.2. Ablation Study

Next, we investigate further the efficacy of each component of the proposed method. Concretely, we conduct ablation experiments to answer the following three questions:

1. *Can the implicit generalization of Bellman operator $\mathbb{T}^\pi$ with PeVFA offer better function approximation?* (Sec.3.1)

2. *Can generalized off-policy evaluation with PeVFA further improve value generalization and learning efficiency?* (Sec.3.2)

3. *Is LPE with dynamic masking an effective method to encode policy network parameters?* (Sec.4)

To answer the first question, we compare the performance difference between $\mathcal{T}^\pi$ with VFA and $\mathbb{T}^\pi$ with PeVFA. We use TD3 and SAC as practical implementations of $\mathcal{T}^\pi$ with VFA, respectively. Correspondingly, we replace only VFA with PeVFA in the TD3 and SAC as practical implementations of $\mathbb{T}^\pi$ with PeVFA, respectively. Note that different from TD3-PeVFA and SAC-PeVFA considered in Sec.5.1, here we do not use the proposed GOPE (Sec.3.2) and DM (Sec.4). The experimental results in Fig.3 show that the Bellman operator $\mathbb{T}^\pi$ with PeVFA outperforms its original counterpart, which empirically demonstrates that implicit generalization of Bellman operator $\mathbb{T}^\pi$ with PeVFA offer

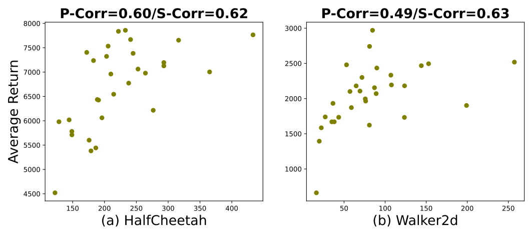

Figure 6. The correlation between the value generalization (x-axis) and the average return (y-axis) of the algorithm based on PeVFA. Each dot represents a trial with a random seed. **Conclusion:** *Pearson* and *Spearman* both show that the weak positive correlation between the above two.

better function approximation.

To answer the second question, we propose two groups of ablation experiments. The first one is **GOPE-P** *vs.* **GOPE-V**. We replace PeVFA in our algorithm (i.e., GOPE-P) with VFA, and the rest of the algorithm is consistent with our algorithm, named **GOPE-V**. In other words, except for the value function approximator, **GOPE-V** retains the generalized off-policy evaluation on the VFA. On the contrary, the other is **GOPE-P** *vs.* **OPE-P**. We retain PeVFA but do not use the generalized off-policy evaluation manner, named **OPE-P**, for which the learning frequency of PeVFA is identical to our algorithm. Besides, we use TD3 and SAC as practical implementations of **OPE-V**. If the performance can be improved by learning the VFA in the generalized off-policy evaluation manner or by learning PeVFA more times, then **GOPE-V** and **OPE-P** should be comparable to our algorithm. Instead, Fig. 4 shows that the performance of **GOPE-V** and **OPE-P** is much lower than the **GOPE-P**. The empirical results show that the generalized off-policy evaluation with PeVFA is crucial to further improve the performance of off-Policy RL algorithm and the two (i.e., GOPE and PeVFA) are complementary.

To answer the third question, we consider five policy representation learning methods in terms of both policy encoder and dynamic masking methods. To highlight the superiority of **LPE**, we compared it with unencoded policy parameters (**Params**) (Faccio et al., 2020) and a multilayer perceptron (**MLP**)-based policy encoder. The policy parameters, as the source data for policy representations, can themselves be used as an uncompressed policy representation. The MLP-based policy encoder flattens the policy into a vector input and utilizes a two-layer feedforward neural network of 256 and 256 hidden nodes. In addition, we also compare the fixed randomly generated policy representation of 64 dimensions, named as **RPR**. **LPE-DM** represents our proposed policy representation learning method which employs both LPE and DM. Fig. 5 reports the experimental results on different methods based on TD3 (Fujimoto et al., 2018) and SAC (Haarnoja et al., 2018), respectively. Obviously, as an original policy representation, the policy parameters are far worse than other policy encoding methods, mainly due to their high-dimensional and highly nonlinear, offering no help in the function approximation. Compared to the MLP-based policy encoder, the most significant characteristic of LPE is that it explicitly considers both the intra-layer and inter-layer structures. LPE-DM shows consistent advantages in the test task, especially in Ant environment.

5.3. On the Correlation between Value Generalization and Learning Performance

In this section, we discuss the correlation between value generalization and learning performance. we first define a *value generalization* metric, Δ based on TD update rule with PeVFA:

$$\Delta = \sum_{t=1}^{T} \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}[\epsilon_{\pi_t} - \epsilon_{\pi_{t+1}}], \tag{16}$$

$$\epsilon_{\pi_t} = \sum_{i=1}^{i=2} (\mathbb{Q}_{\theta_i}(s, a, \chi_{\pi_t}) - y)^2, \tag{17}$$

$$\epsilon_{\pi_{t+1}} = \sum_{i=1}^{i=2} (\mathbb{Q}_{\theta_i}\left(s, a, \chi_{\pi_{t+1}}\right) - y)^2, \tag{18}$$

where $y = r + \gamma \min_{i=1,2} \mathbb{Q}_{\theta_i^-}\left(s', \pi_{t+1}\left(s'\right), \chi_{\pi_{t+1}}\right)$. $\epsilon_{\pi_t} - \epsilon_{\pi_{t+1}}$ measures the difference of fitting TD target y using PeVFA with two adjacent policy representations (i.e., χ_{π_t}, χ_{π_t+1}, marked in blue). Here, we replace only VFA with PeVFA in the TD3 algorithm and use the LPE without DM as the policy encoder. We repeat the experiment 30 trials with different random seeds on HalfCheetah and Walker2d tasks, respectively, and store the evaluation returns and the value generalization metric of each experiment. To increase the reliability of the experimental results, we adopt two correlation coefficients, i.e., *Pearson (P-Corr)* and *Spearman (S-Corr)*. From Fig. 6, we can obtain 1) value generalization Δ is positive (x-axis) (i.e., on the whole, $\epsilon_{\pi_t} \geq \epsilon_{\pi_{t+1}}$), which indicates that PeVFA has the ability of positive generalization. 2) The results of the two correlation coefficients are around 0.6, which indicates that the positive generalization of PeVFA improves learning performance.

6. Conclusion

This paper proposes a general algorithm, Off-policy PeVFA for boosting off-policy RL in terms of efficiency and stability. Off-policy PeVFA adopts the Bellman operator with PeVFA for better function approximation and employs a PeVFA-based generalized off-policy evaluation method to further improve value generalization and learning efficiency. Besides, this work investigates the evolvement of neural nodes of the policy network during the learning process and proposes a policy network representation learning method, LPE-DM. The empirical results demonstrate that our algorithm is general enough to incorporate into other off-policy algorithms.

References

Achiam, J., Knight, E., and Abbeel, P. Towards characterizing divergence in deep q-learning. *arXiv preprint arXiv:1903.08894*, 2019.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Chen, X., Wang, C., Zhou, Z., and Ross, K. W. Randomized ensembled double q-learning: Learning fast without a model. In *ICLR*, 2021.

Degris, T., White, M., and Sutton, R. S. Off-policy actor-critic. *CoRR*, abs/1205.4839, 2012.

Faccio, F., Kirsch, L., and Schmidhuber, J. Parameter-based value functions. *arXiv preprint arXiv:2006.09226*, 2020.

François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., Pineau, J., et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.

Fujimoto, S., Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In *ICML*, pp. 1587–1596. PMLR, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, pp. 1861–1870. PMLR, 2018.

Harb, J., Schaul, T., Precup, D., and Bacon, P. Policy evaluation networks. *arXiv preprint arXiv:2002.11833*, 2020.

Hasselt, H. Double q-learning. *NeurIPS*, 23, 2010.

Kumar, A., Gupta, A., and Levine, S. Discor: Corrective feedback in reinforcement learning via distribution correction. In *NeurIPS*, 2020.

Kuznetsov, A., Shvechikov, P., Grishin, A., and Vetrov, D. P. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *ICML*, volume 119, pp. 5556–5566. PMLR, 2020.

Lan, Q., Pan, Y., Fyshe, A., and White, M. Maxmin q-learning: Controlling the estimation bias of q-learning. In *ICLR*, 2020.

Liang, L., Xu, Y., McAleer, S., Hu, D., Ihler, A., Abbeel, P., and Fox, R. Reducing variance in temporal-difference value estimation via ensemble of deep networks. In *ICML*, volume 162, pp. 13285–13301. PMLR, 2022.

Ota, K., Oiki, T., Jha, D. K., Mariyama, T., and Nikovski, D. Can increasing input dimensionality improve deep reinforcement learning? In *ICML*, volume 119, pp. 7424–7433. PMLR, 2020.

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

Raileanu, R., Goldstein, M., Szlam, A., and Fergus, R. Fast adaptation to new environments via policy-dynamics value functions. In *ICML*, volume 119, pp. 7920–7931. PMLR, 2020.

Saglam, B., Cicek, D. C., Mutlu, F. B., and Kozat, S. S. Off-policy correction for actor-critic algorithms in deep reinforcement learning. *arXiv preprint arXiv:2208.00755*, 2022.

Sang, T., Tang, H., Ma, Y., Hao, J., Zheng, Y., Meng, Z., Li, B., and Wang, Z. Pandr: Fast adaptation to new environments from offline experiences via decoupling policy and environment representations. *arXiv preprint arXiv:2204.02877*, 2022.

Shah, R. M. and Kumar, V. RRL: resnet as representation for reinforcement learning. In *ICML*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9465–9476. PMLR, 2021.

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

Sutton, R. S. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tang, H., Meng, Z., Hao, J., Chen, C., Graves, D., Li, D., Yu, C., Mao, H., Liu, W., Yang, Y., Tao, W., and Wang, L. What about inputing policy in value function: Policy representation and policy-extended value function approximator. *arXiv preprint arXiv:2010.09536*, 2020.

Van Hasselt, H., Guez, A., and Silver, D. Deep reinforcement learning with double q-learning. In *AAAI*, volume 30, 2016.

Van Hasselt, H., Doron, Y., Strub, F., Hessel, M., Sonnerat, N., and Modayil, J. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

A. Related work

Closely related to our work, several works also study the Policy-extended Value Function Approximator for more efficient value estimation. Harb et al. (Harb et al., 2020) propose Policy Evaluation Networks (PVN) to approximate the expected return of different policy π, where the policy representation is obtained through the proposed *Network Fingerprinting*. Raileanu et al. (Raileanu et al., 2020) propose Policy-Dynamics Value Function (PDVF) which takes both policy and task context as additional inputs, for the purpose of value generalization among policies and tasks so that to adapt quickly to new tasks. Faccio et al. (Faccio et al., 2020) propose Parameter-based Value Functions (PVFs) which take policy parameters as inputs. Tang et al. (Tang et al., 2020) propose PPO-PeVFA which learn a representation (low-dimensional embedding) for an RL policy from its network parameters to extend value function. While these works have attempted to obtain better value generalization by extended value function, they consider more on-policy or offline scenarios.

B. Details of the PeVFA-based Clipped Double Q-learning method

In this work, we discuss four variants for PeVFA-based Clipped Double Q-learning implementation. Concretely, we consider the TD3 algorithm based on Clipped Double Q-learning which maintains a pair of actors (π_ϕ, π_{ϕ^-}) and critics $(Q_{\theta_1}, Q_{\theta_2})$. In the first variant, we maintain a pair of policy encoders $(f_\psi(\phi), f_{\psi^-}(\phi))$, taking the parameters ϕ of policy π_ϕ as input. The value approximation of both critics is formulated as: $\mathbb{Q}_{\theta_i}(s,a,f_\psi(\phi)) \leftarrow r+\gamma \min_{i=1,2}\mathbb{Q}_{\theta_i^-}\left(s',\pi_{\phi^-}(s'),f_{\psi^-}(\phi)\right)$. In the second variant, we maintain a pair of policy encoders $(f_\psi(\phi), f_{\psi^-}(\phi^-))$, taking the parameters ϕ of policy π_ϕ and the parameters ϕ^- of policy π_{ϕ^-} as input, respectively. The value approximation of both critics is formulated as: $\mathbb{Q}_{\theta_i}(s,a,f_\psi(\phi)) \leftarrow r+\gamma \min_{i=1,2}\mathbb{Q}_{\theta_i^-}\left(s',\pi_{\phi^-}(s'),f_{\psi^-}(\phi^-)\right)$. In the third variant, we maintain a pair of policy encoders $(f_\psi(\phi^-), f_{\psi^-}(\phi^-))$, taking the parameters ϕ^- of policy π_{ϕ^-} as input. The value approximation of both critics is formulated as: $\mathbb{Q}_{\theta_i}(s,a,f_\psi(\phi^-)) \leftarrow r+\gamma \min_{i=1,2}\mathbb{Q}_{\theta_i^-}\left(s',\pi_{\phi^-}(s'),f_{\psi^-}(\phi^-)\right)$. In the fourth variant, we maintain two pairs of policy encoders $(f_{\psi_{1,2}}(\phi), f_{\psi^-_{1,2}}(\phi))$, taking the parameters ϕ of policy π_ϕ as input. The value approximation of both critics is formulated as: $\mathbb{Q}_{\theta_i}(s,a,f_{\psi_i}(\phi)) \leftarrow r+\gamma \min_{i=1,2}\mathbb{Q}_{\theta_i^-}(s',\pi_{\phi^-}(s'),f_{\psi_i^-}(\phi))$. Table 3 reports the experimental results of the four variants (PeCDQ-v1, PeCDQ-v2, PeCDQ-v3, PeCDQ-v4) as well as the baselines (RA, TD3). The empirical results show the superiority of the first variant which is adopted in our algorithm.

Table 3. Discussion of different implementation variants for PeVFA-based Clipped Double Q-learning. Average evaluation returns and Max evaluation returns (± half a std) over 10 trials for algorithms. The best results are bolded.

Environment	Ave-Evaluation					
	RanP	*TD3*	*PeCDQ-v1*	*PeCDQ-v2*	*PeCDQ-v3*	*PeCDQ-v4*
HalfCheetah	-363.75±83.99	7866.79±546.25	8620.37±110.55	8181.73±405.1	7961.49±310.02	8724.56±138.04
Hopper	21.43±8.23	2464.53±159.18	2544.11±122.45	2588.21±129.02	2427.17±38.0	2536.36±131.46
Walker2d	-7.76±1.67	2573.99±286.84	3271.57±169.02	3147.0±259.95	3302.85±152.21	3425.49±86.56
Ant	926.89±14.53	2265.99±97.2	3034.7±300.83	2998.42±396.9	2645.82±204.38	2497.67±72.48
Norm. Agg.	0	1	**1.24 (↑ 24%)**	1.21 (↑ 21%)	1.14 (↑ 14%)	1.16 (↑ 16%)
Environment	Max-Evaluation					
	RanP	*TD3*	*PeCDQ-v1*	*PeCDQ-v2*	*PeCDQ-v3*	*PeCDQ-v4*
HalfCheetah	-340.6±92.09	9920.17±750.17	10664.98±198.68	10637.38±271.78	10461.47±530.02	11182.97±236.79
Hopper	22.72±8.73	3659.1±28.97	3644.91±29.2	3633.78±36.7	3637.92±49.27	3637.23±35.15
Walker2d	-6.99±2.23	4187.83±287.79	4854.9±226.1	4951.48±229.64	4835.86±228.55	5196.72±159.75
Ant	937.77±15.32	3474.56±219.35	4609.4±549.87	4479.09±436.54	4039.76±519.72	3801.51±412.43
Norm. Agg.	0	1	**1.17 (↑ 17%)**	1.16 (↑ 16%)	1.11 (↑ 11%)	1.12 (↑ 12%)

C. Details of generalized off-policy evaluation with PeVFA

Policy representations endow PeVFA $\mathbb{Q}_\theta$ with the property of generalizing across policies which makes full use of old knowledge to improve efficiency in policy evaluation. However, during the learning process, the knowledge obtained through the value learning of early historical policies may be too old to benefit the generalization of PeVFA across policies. Therefore, we propose to only perform the value learning of the proximal policies of the current policy. To be specific, for each policy update step, we store the updated policy and maintain a proximal policy buffer D of size p. To improve the learning efficiency, in the value learning stage of the historical policies (i.e., Lines 5-15 in Algorithm 3.3), we randomly sample 10 policies at intervals 10 from the policy buffer D for each iteration. With TD3 as the baseline, Table 4 reports the experimental results of different buffer sizes. Compared with the cases of $p = 0$ (i.e., value learning without performing historical policies) and $p = x$ (i.e., value learning with performing all historical policies), the value learning of proximal historical policies ($p = 5000$, $p = 10000$) obtain better results with respect to the Ave-Evaluation, which demonstrates that the advantages of the proposed GOPE in terms of stability and learning efficiency. For all the experiments, we uniformly set $p = 20000$. In the future, we will explore better historical policy sampling methods.

Table 4. Discussion of buffer size for proximal policies. Average evaluation returns and Max evaluation returns (± half a std) over 10 trials for algorithms. $p = x$ indicates sampling from the policy buffer containing all historical policies. The best results are bolded for each environment.

Environment	Ave-Evaluation			
	p=x	*p=20000*	*p=5000*	*p=0*
HalfCheetah	8990.75±133.98	**9048.27±168.56**	8920.27±223.68	8693.77±226.57
Hopper	2758.75±108.01	2778.23±101.82	**2779.54±86.74**	2579.82±38.96
Walker2d	2268.95±371.82	3241.44±194.73	**3496.18±217.25**	3416.55±258.93
Ant	2579.88±717.87	**3606.0±281.72**	3073.98±375.69	3223.43±245.85
Environment	Max-Evaluation			
	p=x	*p=20000*	*p=5000*	*p=0*
HalfCheetah	11152.91±124.74	**11254.0±159.42**	11110.28±118.06	10803.12±447.49
Hopper	3657.07±19.92	3666.16±26.83	3707.71±25.66	**3728.67±40.22**
Walker2d	3980.16±74.25	4819.58±156.07	4881.04±209.69	**5123.22±263.97**
Ant	4253.95±526.31	5203.67±365.05	4424.04±556.08	**5288.47±323.18**

D. Details of Policy Network Representation Learning

In essence, any data characterizing policies can be used as the policy data sources, such as parameters of a policy network, trajectories, etc. However, compared with the trajectories with high randomness, the parameters of the policy network are usually available and highly deterministic, which is more convenient for learning policy representation. This work is based on the actor-critic method which has a separate policy network, thus, we can utilize the parameters of the policy network as the data for learning policy representation. To better characterize policies, we propose a Layer-wise Permutation-invariant Encoder with Dynamic Masking (LPE-DM), denoted by f_ψ, which explicitly considers the characteristics of policy network structure (i.e., the intra-layer and inter-layer structures) and the variation of policy parameters during the learning process. An illustration of LPE-DM is in Fig. 7. The main idea of LPE-DM is to perform permutation-invariant transformation for inner-layer weights and biases for each layer first and then concatenate encoding of layers. In particular, in this work, based on the empirical discovery of the activity of neural nodes of the policy network during the learning process, we characterize the policy using the policy parameters associated with neural nodes with high activity.

E. Hyperparameters

Table 5 shows the common hyperparameters of algorithm used in all our experiments. Table 6 shows the structure of the policy network and PeVFA network for TD3-PeVFA and SAC-PeVFA. To be specific, we utilize a two-layer feed-forward neural network of 64 and 64 hidden units with ReLU activation (except for the output layer) for the policy network. Similarly, the PeVFA network also uses a two-layer feed-forward neural network. As an additional input, the policy representation dimension (pr dim) is set to 64.

F. Pseudo-code of SAC-PeVFA

The pseudo-code of the proposed algorithm, SAC-PeVFA is in Algorithm 2.

G. Complete Learning Curves

Fig. 8 and Fig. 9 shows the learning curves of RanP, TD3, SAC, TD3-PeVFA and SAC-PeVFA corresponding to the results in Table 1 and Table 2.

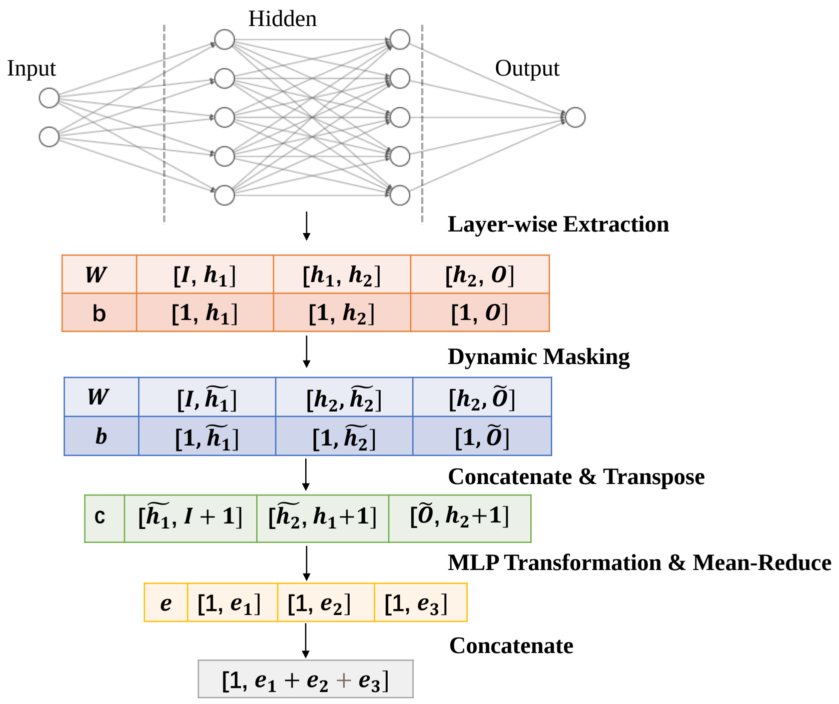
Input
Hidden
Output
Layer-wise Extraction
W | [I, h₁] | [h₁, h₂] | [h₂, O]
b | [1, h₁] | [1, h₂] | [1, O]
Dynamic Masking
W | [I, h̃₁] | [h₂, h̃₂] | [h₂, Õ]
b | [1, h̃₁] | [1, h̃₂] | [1, Õ]
Concatenate & Transpose
c | [h̃₁, I + 1] | [h̃₂, h₁+1] | [Õ, h₂+1]
MLP Transformation & Mean-Reduce
e | [1, e₁] | [1, e₂] | [1, e₃]
Concatenate
[1, e₁ + e₂ + e₃]

Figure 7. An illustration for Layer-wise Permutation-invariant Encoder with Dynamic Masking (LPE-DM). I, h_i and O denote the numbers of units for the input layer, hidden layer and output layer, respectively. $\widetilde{h}_i$ and $\widetilde{O}$ represents the numbers of units using dynamic masking for the corresponding layer.

Table 5. Common hyperparameters. We use '-' to denote the 'not applicable' situation.

Hyperparameters	TD3 & TD3-PeVFA	SAC & SAC-PeVFA
Actor Learning Rate	10^{-3}	3×10^{-4}
Critic Learning Rate	10^{-3}	3×10^{-4}
Target Action Noise	0.2	-
Actor Training Interval	2 steps	1 step
Masking Ratio (η)	0.6	0.6
Discount Factor (γ)	0.99	0.99
Soft Replacement Ratio	0.005	0.002
Replay Buffer Size	200k time steps	200k time steps
Batch Size	100	128
Training Interval	1 step	1 step
Optimizer	Adam	Adam

Algorithm 2 SAC-PeVFA

```
Initialize critic networks $\mathbb{Q}_{\theta_1}$,$\mathbb{Q}_{\theta_2}$, actor network $\pi_\phi$ and policy encoder network $f_\psi$, with random parameters $\theta_1$, $\theta_2$, $\phi$,
$\psi$, target networks ${\theta_1}^- \leftarrow \theta_1$, ${\theta_2}^- \leftarrow \theta_2$, $\psi^- \leftarrow \psi$, replay buffer $\mathcal{B}$, policy buffer $\mathcal{D}$
for iteration $t = 0, 1, 2, \cdots$ do
    Select action $a$ and observe reward $r$ and new state $s'$. Store transition tuple $(s, a, r, s')$ in $\mathcal{B}$. Store policy $\phi$ in $\mathcal{D}$
    Value learning of historical policies
        for $m = 0, 1, 2, \cdots$ do
            Sample mini-batch of $n$ policies from $\mathcal{D}$ and mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
            for $v = 1, 2, \cdots, n$ do
                $a', \log \pi_{\phi_v}(\cdot|s') \leftarrow \pi_{\phi_v}(s')$, store $N$ transition tuple $(s, a, r, s', a', \log \pi_{\phi_v}(\cdot|s'), v)$ in $\mathcal{B}'$
            end for
            Sample mini-batch of $N$ transitions $(s, a, r, s', a', \log \pi_{\phi_v}(\cdot|s'), v)$ from $\mathcal{B}'$
            $y \leftarrow r + \gamma(\min_{i=1,2} \mathbb{Q}_{\theta_i^-}\left(s', a', f_{\psi^-}(\phi_v)\right) - \alpha \log \pi_{\phi_v}(\cdot|s'))$
            Update critics $\theta_i, \psi \leftarrow \operatorname{argmin}_{\theta_i,\psi} \mathbb{E}_{(s,a,r,s',a',\log \pi_{\phi_v}(\cdot|s'),v)\sim\mathcal{B}'} \sum_{i=1}^{i=2} \frac{1}{2}\left(y - \mathbb{Q}_{\theta_i}\left(s, a, f_\psi(\phi_v)\right)\right)^2$
            Update target networks $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$, $\psi' \leftarrow \tau\psi + (1-\tau)\psi'$
        end for
    Value learning of current policy
        for $m = 0, 1, 2, \cdots$ do
            Sample mini-batch of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$
            $y \leftarrow r + \gamma(\min_{i=1,2} \mathbb{Q}_{\theta_i^-}\left(s', \pi_\phi(s'), f_{\psi^-}(\phi)\right) - \alpha \log \pi_\phi(\cdot|s'))$
            Update critics $\theta_i, \psi \leftarrow \operatorname{argmin}_{\theta_i,\psi} \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}} \sum_{i=1}^{i=2} \frac{1}{2}\left(y - \mathbb{Q}_{\theta_i}\left(s, a, f_\psi(\phi)\right)\right)^2$
            Update actor $\phi \leftarrow \operatorname{argmax}_\phi \mathbb{E}_{(s,a,r,s')\sim\mathcal{B}}(\min_{i=1,2} \mathbb{Q}_{\theta_i}\left(s, \pi_\phi(s), f_\psi(\phi)\right) - \alpha \log \pi_\phi(\cdot|s))$
            Update target networks $\theta_i^- \leftarrow \tau\theta_i + (1-\tau)\theta_i^-$, $\psi^- \leftarrow \tau\psi + (1-\tau)\psi^-$
    end for
```

Table 6. Structure of policy network and PeVFA network. We use '-' to denote the 'not applicable' situation.

Method	Layer	Policy Network ($\pi(a\|s)$)	PeVFA Network ($\mathbb{V}(\cdot)$)
TD3-PeVFA	Fully Connected	(state dim, 64)	(state dim + action dim + pr dim, 256)
	Activation	ReLU	ReLU
	Fully Connected	(64, 64)	(256, 256)
	Activation	ReLU	ReLU
	Fully Connected	(64, action dim)	(256, 1)
	Activation	tanh	None
SAC-PeVFA	Fully Connected	(state dim, 64)	(state dim + action dim + pr dim, 256)
	Activation	ReLU	ReLU
	Fully Connected	(64, 64)	(256, 256)
	Activation	ReLU	ReLU
	Fully Connected	(64, action dim)	(256, 1)
	Activation	None	None
	Fully Connected	(64, action dim)	-
	Activation	None	-

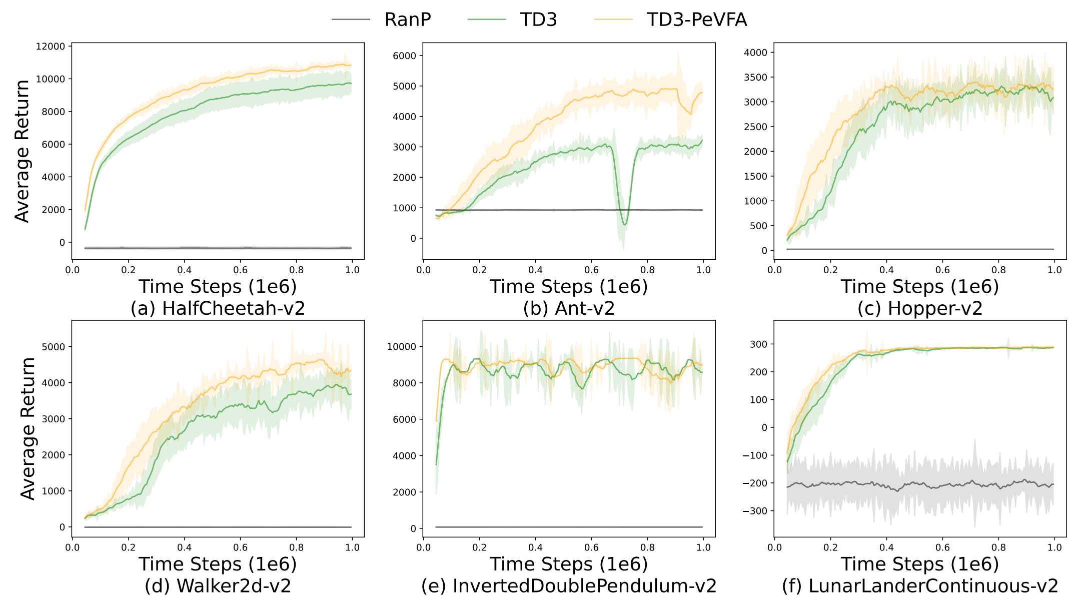

Figure 8. Learning curves for the OpenAI gym continuous control tasks. The shaded region represents half a standard deviation of the average evaluation over 10 trials. Curves are smoothed uniformly for visual clarity.

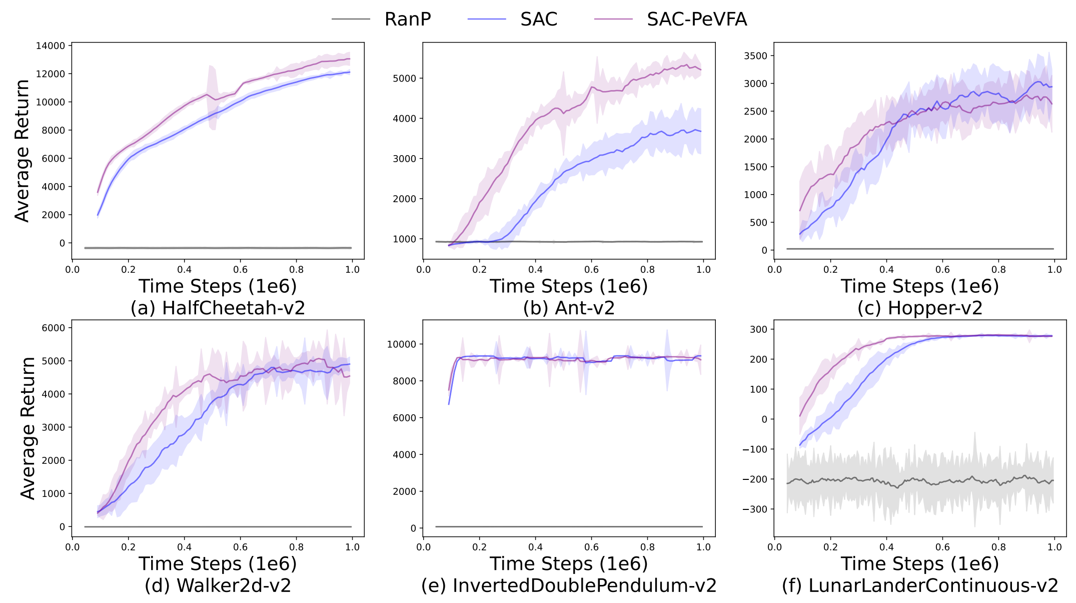

Figure 9. Learning curves for the OpenAI gym continuous control tasks. The shaded region represents half a standard deviation of the average evaluation over 10 trials. Curves are smoothed uniformly for visual clarity.