Representation Gap in Deep Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Deep reinforcement learning gives the promise that an agent learns good policy from high-dimensional information. Whereas representation learning removes irrelevant and redundant information and retains pertinent information. We consider the representation capacity of action value function and theoretically reveal its inherent property, *representation gap* with its target action value function. This representation gap is favorable. However, through illustrative experiments, we show that the representation of action value function grows similarly compared with its target value function, i.e. the undesirable inactivity of the representation gap (*representation overlap*). Representation overlap results in a loss of representation capacity, which further leads to sub-optimal learning performance. To activate the representation gap, we propose a simple but effective framework Policy Optimization from Preventing Representation Overlaps (POPRO), which regularizes the policy evaluation phase through differing the representation of action value function from its target. We also provide the convergence rate guarantee of POPRO. We evaluate POPRO on gym continuous control suites. The empirical results show that POPRO using pixel inputs outperforms or parallels the sample-efficiency of methods that use state-based features.

1 Introduction

By combining representation capabilities of deep neural network (DNN) with the credit assignment capabilities of reinforcement learning (RL), Deep RL (DRL) is able to develop a self-control agent that can perform complex control tasks from high-dimensional observations such as image pixels and sensors information [1, 2, 3], where DNN is utilized to parameterize the policy function or value function. DRL gives the promise that the agent learns good policy by tackling the high-dimensional information. However, this naturally needs to remove irrelevant and redundant information and retain pertinent information, which is the job of representation learning. Thus, representation learning in DRL has attracted much attention from researchers [4, 5].

Representation learning methods in DRL focus on how to obtain good task-related representations [6, 5, 7, 8, 9, 10, 11]. Some of them borrow insights from other areas such as computer vision [10, 11]. Some works attempt to use auxiliary tasks (e.g. predicting the future conditioned on the past observations or actions [12, 13]) to improve the representational capacity of RL and thus improve the empirical performance. These works somehow did not attach the importance of the inherent representation property of the core instrument in DRL, the action value function. In this work, we investigate the representation capacity in action value network and theoretically show that a good action value function representation should have the inherent property of *representation gap*.

Typical DRL methods utilize a target value function to stabilize the policy evaluation phase [2, 14, 15, 16, 17]. We consider the representation capacity of action value function under this setting. Following the commonly used definition of representation of action value function [18, 19, 20, 21], we separate the action value network into a nonlinear encoder and a linear layer. The representation can be considered as the output of the nonlinear encoder. We start by investigating the Bellman equation [22] from the perspective of representation. We then theoretically develop our notion of 'good' representation from the Bellman update of the action value function, i.e., a good representation of action value function should have an inherent *representation gap* from its target value network. We then experimentally check the representation of action value function and its target, finding that the representation of value function and its target grow similarly as training, resulting in an undesirable phenomenon, which we call *representation overlap*. This similarity leads to the collapse of the representation gap. The similarity we catch is not natural, because the input of the corresponding neural network is different state-action pairs, and the action value network is different from its target.

The aforementioned similarity of representation between value function and its target inspired us to improve the representation capacity of the corresponding network by activating the representation gap. Thus, we propose an easy-to-implement and effective framework, Policy Optimization from Preventing Representation Overlaps (POPRO), to activate representation gap and prevent the representation overlap for boosting the performance of DRL algorithms. Specifically, POPRO regularizes the policy evaluation phase by pushing the representation of the value function away from its target, while keeps the policy optimization phase unchanged. We study the representations of the POPRO framework on PyBullet continuous control suite[23], and we find that the representation similarity between the value function and its target has been significantly alleviated. Meanwhile, the empirical performance of POPRO outperforms other tested algorithms such as TD3[15], METD3 [24], etc. We then extend POPRO to high dimensional input scenarios, DMControl suite [25]. The empirical performance of POPRO outperforms or matches the compared baselines such as CURL [5], DREAMER [7], etc. POPRO using pixel inputs outperforms or parallels the sample-efficiency of methods that use state-based features. We show that the DRL algorithm can be significantly improved by activating the representation gap between the action value function and its target.

In this work, we make the following contributions. (i) We theoretically show that there should exist a *representation gap* between action value function and its target. (ii) We define *representation overlap* phenomenon as representations (of the value function and its target) tend to grow similar when training the value function. (iii) To activate the representation gap, we propose an easy-to-implement and effective framework POPRO which adds a regularizer to penalize the representation overlap phenomenon. In addition, we also provide the convergence rate guarantee of POPRO. (iv) To demonstrate the effectiveness of the POPRO framework, we evaluate it against PyBullet and DMControl suites. The empirical results show that POPRO outperforms or matches the state-of-the-art representation learning RL methods.

2   Related Work

Deep Q-networks [2] approximating state-action value by neural networks, where the optimization objective origins from dynamics programming [22]. And the target network used in DQN [2] laid the foundation for the success of subsequent DRL algorithms [26, 27, 14, 15, 16, 28, 29, 24, 30, 31, 17]. It is generally acknowledged that good representations are conducive to improve the performance of RL[32]. There are various notions of representations, including representations of observations [5, 33], representations of the dynamics model [34, 35, 36, 37], and representations of policies[38]. Recent works [39, 40, 41, 5, 42] used self-supervised, unsupervised and contrastive representation learning approaches to improve the performance of deep RL algorithms.

Most prior works of representation learning use auxiliary tasks or mutual-information based representation learning. The UNREAL algorithm [39] added unsupervised auxiliary tasks to conventional deep RL methods. The PBL agent [40] achieved good performance in simulation settings by adding an auxiliary optimization term to objectives that utilize the forward and backward prediction history

information of neural networks. The CPC [41] algorithm introduced contrastive loss in time level and achieve good performance in the simulator. CURL algorithm [5] introduces multiple auxiliary tasks including augmentation, contrastive learning.

Recently, there emerged several works studying representation learning from a geometric view[43, 44, 38]. [43, 44] considered geometry of value functions, while [38] studies the geometry of policy functions.

However, there are limited prior works that explicitly consider representation capacity loss [45], which imposes regularization to force the network to converge to the original weights. Our work differentiates from previous works from the following three perspectives. First, our work origins from analyzing the inherent representation capacity of action value function and its target. Our insights are from activating the representation gap. Our proposed framework POPRO explicitly consider the representation capacity of action value network instead of considering how to learn good representations with the help of auxiliary tasks. We also provide the convergence rate guarantee of POPRO. Second, POPRO can couple with other algorithms adopting complex auxiliary unsupervised, self-supervised, and contrastive learning tasks. What's more, the experimental results show that POPRO is parallel to the method utilizing auxiliary tasks. The proposed framework POPRO is markedly clear and easy to implement. Third, our algorithm is also suitable for pixel input and vector input, which is verified in experiments.

3 Background

In this paper, we formulate RL as a Markov Decision Process (MDP) with a six-tuple $< S, \mathcal{A}, \mathcal{R}, p, \rho_0, \gamma >$, where S is state space, $\mathcal{A}$ is action space, $\mathcal{R} : S \times \mathcal{A} \to \mathbb{R}$ is a scalar reward function, $p(s'|s, a)$ is transition probability function, ρ_0 is the initial state distribution, and $\gamma \in (0, 1)$ is the discount factor determining the rate of decay of importance rewards. At each time step t, the agent encounters state s_t and chooses an action a w.r.t. its policy function π, deterministic or stochastic, then encounters a new state s_{t+1} and a reward r_t. RL aims to optimize the policy through return, which is defined as $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$. Action value function $Q^\pi(s, a)$ represents the quality of a specific action a in a state s. Formally, action value (Q) function is defined as

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi, p}[R_\tau | s_0 = s, a_0 = a], \quad (1)$$

where trajectory τ is a state-action sequence $(s_0, a_0, s_1, a_1, s_2, a_2 \cdots)$ induced by policy π
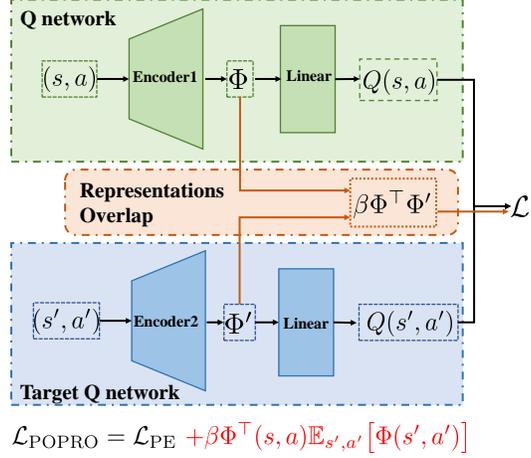

Algorithm 1 POPRO framework
Initialize actor network π, and critic network Q with random parameters
Initialize target networks and replay buffer $\mathcal{B}$
Initialize β, total steps T, and $t = 0$
Reset the environment and receive initial state s
while $t < T$ **do**
    Select action w.r.t. its policy π and receive reward r, new state s'
    Store transition tuple (s, a, r, s') to $\mathcal{B}$
    Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
    Compute target value y and target representation ϕ
    Update critic by minimizing eq. (8)
    Update actor
    Update target networks:
    $t \leftarrow t + 1$
    $s \leftarrow s'$
end while

---

and transition probability function p. A four-tuple (s_t, a_t, r_t, s_{t+1}) is called a transition. The Q value can be recursively computed by Bellman equation [22]

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s', a'}[Q^\pi(s', a')], \quad (2)$$

where $s' \sim p(\cdot|s, a)$ and $a' \sim \pi(s)$. The process of evaluating value function is known as policy evaluation phase.

When the action space is very large or continuous, indirectly obtaining the policy by action value function is intractable as DQN does [46]. Thus, Policy Gradient theorem [47] is introduced to optimize the policy directly

$$\mathcal{J}(\pi) = \mathbb{E}_{\tau \sim \pi, p} \left[\sum_{t=0}^{T} \nabla_\phi \log \pi(a_t \mid s_t; \phi) R(\tau) \right]. \quad (3)$$

3

However, DRL suffers from unstable training issue. To stabilize the training of DRL, DQN [2] introduced a target network to update the parameters of network with $\theta' \leftarrow \eta\theta + (1 - \eta)\theta'$, where η is a small constant controlling the update scale. θ is the parameters of Q network. And θ' denotes exponential moving average (EMA) of θ.

4 Policy Optimization from Preventing Representation Overlap

In this section, we first theoretically define the representation gap. Then we experimentally show the phenomenon that the representation of action value function would grow similar to that of its target. To activate the representation gap, we propose Policy Optimization from Preventing Representation Overlaps (POPRO) framework, which regularizes the policy evaluation phase through differing the representation of action value function from its target. We also provide the convergence rate guarantee of POPRO.

$$\mathcal{L}_{\text{POPRO}} = \mathcal{L}_{\text{PE}} + \beta\Phi^\top(s, a)\mathbb{E}_{s', a'}\left[\Phi(s', a')\right]$$

Figure 1: The structure of the POPRO framework. the Encoder is a nonlinear operator, and the state action pairs generate the representation Φ through the encoder and then the action value through a linear layer. POPRO regularizes the policy evaluation phase by differing the representation Φ of the action value network from its target when training. $\mathcal{L}_{\text{PE}}$ is an action value loss function. And β is a small positive constant, controlling the magnitude of the regularization effectiveness.

4.1 Representation gap

We define the representation of value function to facilitate subsequent discussions.

Definition 4.1 (Representation of action value function). Given a multilayer neural network representing Q function parameterized by Θ, Θ_i represents the parameters of ith layer, Θ_{-1} represents the parameters of the last layer, and Θ_+ represents the parameters of the neural networks except for those of the last layer. Then the representation Φ of Q function is defined as

$$Q(s, a; \Theta) = \langle \Phi(s, a; \Theta_+), \Theta_{-1}\rangle \tag{4}$$

An intuitive way to understand the representation of action value function is that we can split the action value network as a nonlinear encoder and a linear part. The representation of action value function is the output of the nonlinear encoder.

Definition 4.2 (Representation Gap). Given a multi-layer neural network representing Q function parameterized by Θ, Θ_i represents the parameters of ith layer, Θ_{-1} represents the parameters of the last layer, and Θ_+ represents the parameters of the neural networks except for those of the last layer. Then the representation gap is defined as

$$\Delta\Phi(s, a) = \Phi(s, a; \Theta_+)^\top - \gamma\mathbb{E}_{s', a'}\Phi(s', a'; \Theta'_+)^\top \tag{5}$$

Theorem 4.3 (Size of Representation Gap). *There exists a representation gap after the policy evaluation phase converges. The representation gap $\Delta\Phi(s, a) = \Phi(s, a; \Theta_+)^\top - \gamma\mathbb{E}_{s', a'}\Phi(s', a'; \Theta'_+)^\top$ satisfies*

$$\|(\Phi(s, a; \Theta_+)^\top - \gamma\mathbb{E}_{s', a'}\Phi(s', a'; \Theta'_+)^\top)\| \geq \frac{r(s, a)}{\|\Theta_{-1}\|} \tag{6}$$

4

171 *Proof.* Check the Appendix section 7 for the proof. □

172 The theorem 4.3 shows that there exists an inherent representation gap between the action value
173 network and its target after the policy evaluation process converges. The Representation gap is natural,
174 because the input of the corresponding neural network is different state-action pairs, and the action
175 value network is different from its target.
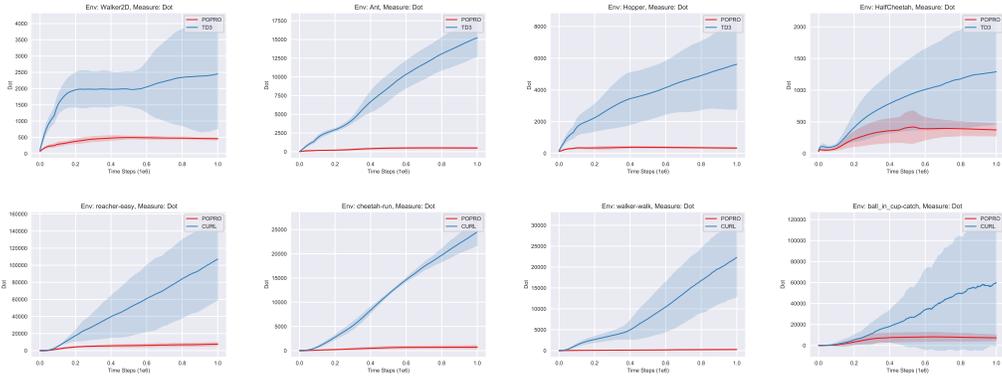
4.2 Collapse of representation gap

Figure 2: Similarity measures for representation of action value functions of TD3 and POPRO agents.
The shaded area stands for a standard deviation. Column: various environments. Row: different
algorithm. The representations of action value networks of TD3 and CURL agents grow similar
as training processing, which results in the collapse of representation gap. But POPRO framework
does not. We present experiments under Manhattan distance and cosine similarity measures in the
Appendix fig. 5

177 The theorem 4.3 shows that there exists a representation gap between the action value function and
178 its target. Thus, we experimentally check the representation gap between action value function and
179 its target in two algorithm TD3 and CURL.

180 Firstly, we choose three computationally easy similarity measures to evaluate the representation gap.
181 Specifically, we take normalized Manhattan distance $M(\Phi_1, \Phi_2) = \frac{1}{n}\|\Phi_1 - \Phi_2\|_1$ where n is the
182 dimension of Φ_1, which can measure the distance in each dimension. We also choose cosine similarity
183 $Cosine(\Phi_1, \Phi_2) = \Phi_1^\top \Phi_2 / (\|\Phi_1\|_2 \|\Phi_2\|_2)$, which measures the similarity of the representation as a
184 whole. We also measure the dot product of two representations, defined by $Dot(\Phi_1, \Phi_2) = \langle \Phi_1, \Phi_2 \rangle$.

185 Then we train TD3 [15] and CURL [5] agents (coupled with SAC [16]) on PyBullet [23] and
186 DMControl suites, interacting with Gym [48] protocol.

187 We show the experimental results in fig. 2, which shows that the representation of action value network
188 grows similarly with its target when training. This similarity results in *representation overlap*. The
189 similarity we catch is not natural, because the input of the corresponding neural network is different
190 state-action pairs, and the action value network is different from its target. Representation overlap
191 leads to the inactivity of the representation gap. The two inputs pairs $(s, a), (s', a')$ of action value
192 network and its target satisfy the dynamics of environments $p(s'|s, a)$ and policy $\pi(\cdot|\cdot)$. The two
193 input pairs are contiguous in time. The replay buffer setting used by Deep Q-networks [46] is believed
194 to break the correlation of the data at the trajectory level. However, we feed two slightly different
195 neural networks with two adjacent inputs related (s, a) and (s', a') respectively. Thus, the correlation
196 at transition level is naturally kept, which is hard to be broken up. This is a potential reason why the
197 representation overlap happens.

5

4.3 Activating representation gap

Thus, we propose an easy-to-implement and effective framework, Policy Optimization from Preventing Representation Overlaps (POPRO), to activate representation gap and prevent the representation overlap in order to improve the performance of deep RL algorithms. Specifically, POPRO regularizes the policy evaluation phase by keeping the representation of the value function different from its target. For the policy optimization phase, POPRO keeps the conventional policy optimization method.

Section 4.2 shows that in experiments, there exists representation overlap between action value function and its target. The representation of action value function grow similarly compared with the target action value function. However, section 4.1 shows that the representation gap between action value function and its target is inherent. Thus, to improve the representation capacity of action value function, the agent needs to activate the representation gap. Thus, to activate the representation gap, we propose Policy Optimization from Preventing Representation Overlaps (POPRO) framework. In the policy evaluation phase, POPRO regularizes the action value network by keeping the its representation different from its target network. For conventional policy evaluation phase [14, 15, 17], the optimization objective is

$$\mathcal{L}_{\text{PE}}(\Theta) = \left[Q(s,a) - \left(r(s,a) + \gamma \mathbb{E}_{s',a'} \left[Q(s',a';\Theta') \right] \right) \right]^2. \tag{7}$$

For the policy evaluation phase, the optimization objective of POPRO is

$$\mathcal{L}_{\text{POPRO}}(\Theta) = \mathcal{L}_{\text{PE}} + \beta \Phi^\top(s,a;\Theta_+) \mathbb{E}_{s',a'} \left[\Phi(s',a';\Theta'_+) \right], \tag{8}$$

where β is a hyper-parameter controlling the regularization effect of activating representation gap and simultaneously preventing representation overlap. For the policy improvement phase, POPRO can adopt any conventional policy gradient method such as deterministic policy gradient [49], soft policy improvement [16], depending on the specifics implementation. We summarize the POPRO framework in algorithm 1. The POPRO framework can be extended to DRL methods which include policy evaluation phase, such as TD3 [15], SAC [16], etc.

We also provide a theoretical guarantee of the convergence of our algorithm by theorem 4.5.

Assumption 4.4. The l_2-norm of is uniformly bounded by the square of some positive constant G, i.e. $\|\Phi(X;\Theta_+)\|^2 \le G^2$ for any $X \in \mathcal{S} \times \mathcal{A}$ and network weights Θ.

Let T be the Bellman Operator. We have the following convergence result for the core update step in POPRO.

Theorem 4.5 (One-step Approximation Error of POPRO Update). *Suppose assumption 4.4 hold, let $\mathcal{F} \subset \mathcal{B}(\mathcal{S} \times \mathcal{A})$ be a class of measurable function on $\times$ that are bounded by $V_{\max} = R_{\max}/(1-\gamma)$, and let σ be a probability distribution on $\mathcal{S} \times \mathcal{A}$. Also, let $\{(S_i, A_i)\}_{i \in [n]}$ be n i.i.d. random variables in $\mathcal{S} \times \mathcal{A}$ following σ. For each $i \in [n]$, let R_i and S_i be the reward and the next state corresponding to (s_i, a_i). In addition, for $Q \in \mathcal{F}$, we define $Y_i = R_i + \gamma \cdot \max_{a \in \mathcal{A}} Q(S'_i, a)$. Based on $\{(X_i, A_i, Y_i)\}_{i \in [n]}$, we define $\hat{O}$ as the solution to the lease-square with regularization problem,*

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} [f(S_i, A_i) - Y_i]^2 + \beta \Phi(s,a;\Theta) \mathbb{E} \Phi_{s',a'}(s',a';\Theta') \tag{9}$$

Meanwhile, for any $\delta > 0$, let $\mathcal{N}(\delta, \mathcal{F}, \|\cdot\|_\infty)$ be the minimal δ-covering set of $\mathcal{F}$ with respect to l_∞-norm, and we denote by N_δ its cardinality. Then for any $\epsilon \in (0,1]$ and any $\delta > 0$, we have

$$\|\hat{O} - TQ\|_\sigma^2 \le (1+\epsilon)^2 \cdot \omega(\mathcal{F}) + C \cdot V_{\max}^2/(n \cdot \epsilon) + C' \cdot V_{\max} \cdot \delta + 2\beta \cdot G^2, \tag{10}$$

where C and C' are two absolute constants and is defined as

$$\omega(\mathcal{F}) = \sup_{g \in \mathcal{F}} \inf_{f \in \mathcal{F}} \|f - Tg\|_\sigma. \tag{11}$$

We defer the proof of this theorem to Appendix section 7. We follow the proof style of [50].

6

5 Experiments

Table 1: Scores achieved by POPRO (mean and standard deviation over 10 random seeds) on DMControl continuous control suite. The POPRO framework achieves superior performance on the majority (**9** out of **12**) tasks.

500K Step Scores	POPRO	DrQ	CURL	PlaNet	Dreamer	SAC+AE	State SAC
Finger,Spin	871 ± 157	938 ± 103	926 ± 45	561 ± 284	796 ± 183	884 ± 128	923 ± 21
Cartpole,Swingup	850 ± 23	868 ± 10	841 ± 45	475 ± 71	762 ± 27	735 ± 63	848 ± 15
Reacher,Easy	**980 ± 4**	942 ± 71	929 ± 44	210 ± 390	793 ± 164	627 ± 58	923 ± 24
Cheetah,run	**708 ± 20**	660 ± 96	518 ± 28	305 ± 131	570 ± 253	550 ± 34	795 ± 30
Walker,Walk	**958 ± 6**	921 ± 45	902 ± 43	351 ± 58	897 ± 49	847 ± 48	948 ± 54
Ball in cup,Catch	**971 ± 5**	963 ± 9	959 ± 27	460 ± 380	879 ± 87	794 ± 58	974 ± 33
100K Step Scores							
Finger,Spin	851 ± 167	901 ± 104	767 ± 56	136 ± 216	341 ± 70	740 ± 64	811 ± 46
Cartpole,Swingup	**847 ± 24**	759 ± 92	582 ± 146	297 ± 39	326 ± 27	311 ± 11	835 ± 22
Reacher,Easy	**970 ± 5**	601 ± 213	538 ± 233	20 ± 50	314 ± 155	274 ± 14	746 ± 25
Cheetah,run	**441 ± 65**	344 ± 67	299 ± 48	138 ± 88	235 ± 137	267 ± 24	616 ± 18
Walker,Walk	**843 ± 73**	612 ± 164	403 ± 24	224 ± 48	277 ± 12	394 ± 22	891 ± 82
Ball in cup,Catch	**959 ± 7**	913 ± 53	769 ± 43	0 ± 0	246 ± 174	391 ± 82	746 ± 91

We evaluate (i) **performance**: the performance of POPRO through measuring its average return, and (ii) **sample efficiency**: the sample efficiency of POPRO through comparing POPRO with other algorithms at fixed timesteps. Specifically, We couple POPRO framework with TD3 [15] and CURL [5], and conduct experiments on PyBullet [23] and DMControl suites [25]. The proposed algorithm POPRO is simple and easy to implement. Considering the recent concerns of reproducing crisis [51, 52], we do not add any engineering tricks to the implementation of POPRO so that POPRO achieves the purpose as we initially designed.

The reason why we did not conduct an ablation analysis is because the POPRO framework only adds a regularization term to its backbone algorithm. Thus, the comparison with its backbone algorithm naturally becomes an ablation experiment.

5.1 Experimental settings

Random seeds. For the random seeds, if not otherwise specified, we evaluate each tested algorithm over 10 random seeds to ensure the reproducibility of our experiments. Also, we set all seeds fixed in our experiments including but not limited to those used in PyTorch, NumPy, Gym, and CUDA.

Environments. For the experiment environments, we use state-based PyBullet [23] and pixel-based DMControl [25] suites to measure the performance and sample-efficiency of POPRO. We can check the representation capacity of POPRO on state-based and pixel-based suites. Control tasks in PyBullet are generally considered harder than that of MuJoCo [53] suite [54]. For the interactive protocol, we utilize Gym [48] environment. On the PyBullet suite, We run each tested algorithm 1 million timesteps. And every 5k timesteps, we evaluate the average return of the tested algorithm over ten episodes. For DMControl, following CURL experimental setting [5], we measure the performance and sample-efficiency of tested algorithms at 100k and 500k environment timesteps, resulting in DMControl100k and DMControl500k settings.

Baselines. We first evaluate the POPRO framework on the state-based PyBullet suite. We choose TD3, SAC [16], TRPO [28], PPO[29] as our baselines for their superior performance. And we couple POPRO framework with TD3 [15] algorithm in PyBullet experiments. POPRO framework is proposed to prevent the similarity between action value network and its target. Dropout operator [55, 56, 57] is generally believed to prevent feature co-adaptation, which is similar to what POPRO achieves. MEPG utilizes a dropout operator simultaneously acting on the action value network and its target. Thus, we use MEPG framework [24] coupled with the TD3 algorithm as a baseline.

7

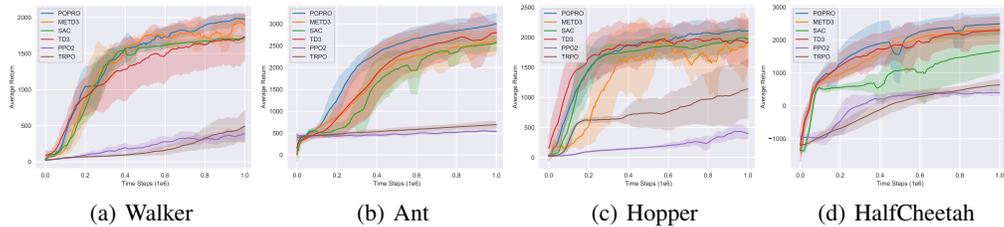| (a) Walker | (b) Ant | (c) Hopper | (d) HalfCheetah |

Figure 3: Performance curves for OpenAI gym continuous control tasks in PyBullet suite. The shaded region represents a standard deviation of the average evaluation over 10 seeds. The curves are smoothed by moving average.

Table 2: The average return of the last ten evaluations over ten random seeds. The maximum average returns are bolded. POPRO outperforms all the algorithms.

1M Step Scores	POPRO	TD3	METD3	SAC	PPO2	TRPO
Ant	**3003 ± 204**	2731 ± 278	2601 ± 246	2561 ± 146	539 ± 25	693 ± 74
HalfCheetah	**2494 ± 276**	2359 ± 229	2345 ± 151	1675 ± 567	397 ± 63	639 ± 154
Hopper	**2106 ± 164**	1798 ± 471	1929 ± 351	1984 ± 103	403 ± 70	1140 ± 469
Walker2D	**1966 ± 58**	1646 ± 314	1901 ± 111	1716 ± 30	390 ± 106	496 ± 206

For the implementation of TD3, we use the authors' implementation. For SAC, we utilize the public implementation [58]. As for TRPO and PPO, we use the OpenAI Baselines [59] codebase. We take the default hyper-parameters as the authors described.

We also evaluate POPRO framework on the pixel-based DMControl suite. We couple POPRO framework with CURL algorithm [5]. Similar to CURL setting [5], we choose (i) PlaNet [6] and (ii) Dreamer [7] both of which learn a world model in latent space and explicitly executing plan; (iii) SAC+AE [8] utilizing VAE [60] and regularized encoder; (iv) CURL utilizing contrastive unsupervised learning to extract high level features from raw pixels; (v) DrQ [9] adopt data augmentation technique; and state-based SAC [16]. Our implementation of POPRO coupled with CURL is based on the CURL codebase.

5.2 Results

Throughout the paper, our proposed framework POPRO uses only one hyperparameter β, controlling the magnitude of the regularization effect. And all the experiments are reported based on $\beta = 5e - 4$. The reader can obtain better performance improvement by choosing a hyper-parameter beta that is more suitable for a specific environment.

PyBullet Suite. To validate the empirical performance of the POPRO framework, we firstly evaluate our proposed algorithm in the state-based suite PyBullet. We show the performance curves of experimental results in fig. 3, and the final 10 evaluation average return in table 2. The empirical performance of POPRO coupled with TD3 outperforms TD3 in all environments. Furthermore, POPRO also surpasses all other tested algorithms, which shows the superior superior performance of POPRO.

DMControl Suite. We then conduct experiments on the DMControl suite. Specifically, we couple POPRO with the CURL algorithm, and run it under DMControl500k and DMControl100k settings. We show the results in table 1. The key findings are as follows: (i) On DMControll500k and DMControl100k settings, POPRO coupled with CURL outperforms its backbone by a large margin on 11 out of 12 tasks, which shows the proposed framework does improve the performance of the coupled algorithm. And the performance improvement on DMC500k shows that the POPRO framework is comparable to the contrastive unsupervised learning methods. (ii) The POPRO framework outper-

8

forms all the tested pixel-based algorithms on most DMControl (**9 out of 12**) tasks. (i) and (ii) show that the sample-efficiency of POPRO framework is superior. (iii) What's more, in section 5.2, we computed the average score of the tested algorithm on all environments under one specific DMControl setting, normalized by the score of State SAC. The results in DMControl100k show that POPRO is more sample-efficient than State SAC and other algorithms. And in the DMControl500k suite, the sample efficiency of POPRO matches that of State SAC. These results illustrate that POPRO greatly improves the empirical performance of the coupled algorithm (TD3 and CURL) by activating the representation gap and improving the algorithm's representation capacity.

5.3 Representation gap of POPRO

To validate whether our framework POPRO achieves our goal or not, we measure the similarity of the action value network and its target in POPRO following section 4.2. We show the experimental results in fig. 2. The representation of the action value network of the POPRO framework does not grow differently compared with the backbone algorithms TD3 and CURL, which verifies that POPRO does help activate the representation gap. Combined with the performance evaluation experiments (section 5.2), our algorithm does outperform the backbone algorithms by improving the representation capacity of action value network through activating the representation gap. Check more experimental results in the Appendix.
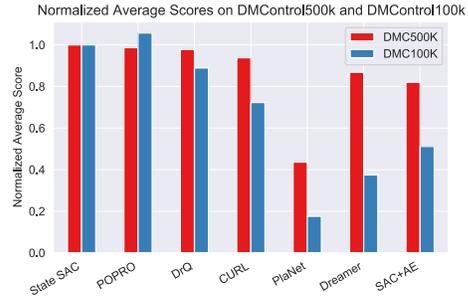

Figure 4: The normalized average scores on the DMControl suite. We normalize the average score of the tested algorithm by the average scores of State SAC. On the DMC100k benchmark, POPRO outperforms all the tested algorithms including State SAC. And POPRO shows comparable performance with other tested algorithms such as DrQ, CURL and etc.

6 Limitations and Conclusion

We identify three limitations of this paper. First, we did not reveal the relationship between the representation gap and other contrastive unsupervised representations for RL. The experiments suggest that the two classes of methods might be orthogonal but in-depth investigations are required. Second, the performance of our POPRO framework matches that of State SAC. However, it is of great intellectual interest to further analyze why this happens. Third, in this work, we only carried out experiments to couple POPRO with TD3 and CURL algorithms. It remains to be checked how POPRO performs when coupled with more DRL algorithms that utilizing auxiliary tasks. Addressing these limitations are meaningful future directions.

In this work, we started by investigating the representations of the action value function and its target and found that there should exist an inherent representation gap between them. We then experimentally check the similarity between action value network and its target for TD3 and CURL methods. The results show that the representation of the action value network grows similar to its target, resulting in representation overlap. To activate the representation gap, we proposed the POPRO framework and provide the convergence rate guarantee of POPRO. We conduct extensive experiments to measure the performance and sample-efficiency of the POPRO framework, where POPRO shows superior performance compared to the tested algorithms. We believe our work sheds light on the nature of the inherent representation gap resulting from combining parameterization tool deep neural network and reinforcement learning.

References

[1] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[3] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[4] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Franois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.

[5] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[6] Danijar Hafner, Timothy P. Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565. PMLR, 2019.

[7] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[8] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 10674–10681. AAAI Press, 2021.

[9] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[10] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 15084–15097, 2021.

[11] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 1273–1286, 2021.

[12] Jiirgen Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments. *Technical Report*, 1990.

[13] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[14] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

[15] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1587–1596. PMLR, 2018.

[16] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[17] Qiang He and Xinwen Hou. Wd3: Taming the estimation bias in deep reinforcement learning. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 391–398. IEEE, 2020.

[18] Justin A. Boyan. Least-squares temporal difference learning. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 49–56. Morgan Kaufmann, 1999.

[19] Nir Levine, Tom Zahavy, Daniel J. Mankowitz, Aviv Tamar, and Shie Mannor. Shallow updates for deep reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3135–3145, 2017.

[20] Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G. Bellemare, and David Silver. The value-improvement path: Towards better representations for reinforcement learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 7160–7168. AAAI Press, 2021.

[21] Marc G. Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taïga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4360–4371, 2019.

[22] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[23] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2021.

[24] Qiang He, Chen Gong, Yuxun Qu, Xiaoyu Chen, Xinwen Hou, and Yu Liu. Mepg: A minimalist ensemble policy gradient framework for deep reinforcement learning. *arXiv preprint arXiv:2109.10552*, 2021.

[25] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm-control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[26] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[27] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

[28] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[29] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[30] Qiang He, Xinwen Hou, and Yu Liu. Popo: Pessimistic offline policy optimization. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4008–4012. IEEE, 2022.

[31] Chen Gong, Qiang He, Yunpeng Bai, Xiaoyu Chen, Xinwen Hou, Yu Liu, and Guoliang Fan. Combing policy evaluation and policy improvement in a unified f-divergence framework. *arXiv preprint arXiv:2109.11867*, 2021.

[32] Dibya Ghosh and Marc G Bellemare. Representations for stable off-policy reinforcement learning. In *International Conference on Machine Learning*, pages 3556–3565. PMLR, 2020.

[33] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[34] Wilson Yan, Ashwin Vangipuram, Pieter Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. *arXiv preprint arXiv:2003.05436*, 2020.

[35] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[36] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[37] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

[38] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. The information geometry of unsupervised reinforcement learning. *arXiv preprint arXiv:2110.02719*, 2021.

[39] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[40] Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap latent-predictive representations for multitask reinforcement learning. In *International Conference on Machine Learning*, pages 3875–3886. PMLR, 2020.

[41] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[42] Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *International Conference on Artificial Intelligence and Statistics*, pages 1–9. PMLR, 2021.

[43] Marc Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[44] Robert Dadashi, Adrien Ali Taiga, Nicolas Le Roux, Dale Schuurmans, and Marc G Bellemare. The value function polytope in reinforcement learning. In *International Conference on Machine Learning*, pages 1486–1495. PMLR, 2019.

[45] Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *Deep RL Workshop NeurIPS 2021*, 2021.

[46] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[47] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[48] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[49] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014.

[50] Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning. In *Learning for Dynamics and Control*, pages 486–489. PMLR, 2020.

[51] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. In *ICLR 2021-Ninth International Conference on Learning Representations*, 2021.

[52] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[53] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, pages 5026–5033. IEEE, 2012.

[54] Antonin Raffin and Freek Stulp. Generalized state-dependent exploration for deep reinforcement learning in robotics. *CoRR*, abs/2005.05719, 2020.

[55] David Warde-Farley, Ian J. Goodfellow, Aaron Courville, and Yoshua Bengio. An empirical analysis of dropout in piecewise linear networks.

[56] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[57] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[58] Denis Yarats and Ilya Kostrikov. Soft actor-critic (sac) implementation in pytorch. https://github.com/denisyarats/pytorch_sac, 2020.

[59] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017.

[60] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[61] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[62] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[63] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015.

Checklist

1. For all authors...

    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] **The main claims made in the abstract and introduction: (i) we theoretically show that there should exist a representation gap between action value function and its target, reflected in theorem 4.3; (2) We demonstrate representation overlap phenomenon, as in fig. 2; (3) We propose an easy-to-implement and effective framework POPRO which adds a regularizer to penalize the representation overlap phenomenon, as in algorithm 1.**

    (b) Did you describe the limitations of your work? [Yes] **Limitations of our work are described in section 6.**

    (c) Did you discuss any potential negative societal impacts of your work? [Yes] **Since our work focuses on the inherent representation property of Deep Reinforcement Learning, we believe it will not result in any negative societal impacts.**

    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] **We have carefully read the ethics review guidelines.**

2. If you are including theoretical results...

    (a) Did you state the full set of assumptions of all theoretical results? [Yes] **The assumption is stated as in assumption 4.4.**

    (b) Did you include complete proofs of all theoretical results? [Yes] **The proof of theorem 4.3 and theorem 4.5 are both provided in Appendix section 7.**

3. If you ran experiments...

    (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] **The code, data, and instructions needed to reproduce the main experimental results are provided in Supplementary Material.**

    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] **Some details are provided in section 5.1. Other training details are provided in Supplementary Material.**

    (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] **The error bars are reported as shaded areas in our experiments.**

    (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] **The total amount of compute and the type of resources used are specified in Supplementary Material.**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [N/A] **Not using any existing assets.**

   (b) Did you mention the license of the assets? [N/A]

   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] **Neither using crowdsourcing nor conducted research with human subjects.**

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

7 Appendix A: Proofs

Theorem 7.1 (Size of Representation Gap). *There exists a representation gap after the policy evaluation phase converging. The representation gap $\Delta\Phi(s,a) = \Phi(s,a;\Theta_+)^\top - \gamma\mathbb{E}_{s',a'}\Phi(s',a';\Theta'_+)^\top$ satisfies $\|(\Phi(s,a;\Theta_+)^\top - \gamma\mathbb{E}_{s',a'}\Phi(s',a';\Theta'_+)^\top)\| \geq \frac{r(s,a)}{\|\Theta_{-1}\|}$*

Proof. Following eq. (4), the Bellman Equation eq. (2) can be rewritten as

$$
\begin{aligned}
\Phi(s,a;\theta_+)^\top\theta_{-1} &\leftarrow r(s,a) + \gamma\frac{1}{N}\sum_{i=1}^N\Phi(s'_i,\pi'(a'_i);\theta'_+)^\top\theta'_{-1} \\
&= \phi(s,a;\Theta_+)^\top\Theta_{-1} \leftarrow r(s,a) + \gamma\mathbb{E}_{s',a'}\phi(s',a';\Theta'_+)^\top\Theta'_{-1},
\end{aligned}
\tag{12}
$$

After the policy evaluation converges, the Θ, Θ' satisfy $\mathbb{E}\Theta = \mathbb{E}\Theta'$.

$$
\begin{aligned}
(\phi(s,a;\Theta_+)^\top - \gamma\mathbb{E}_{s',a'}\Phi(s',a';\Theta'_+)^\top)\Theta_{-1} &= r(s,a), \\
\|(\Phi(s,a;\Theta_+)^\top - \gamma\mathbb{E}_{s',a'}\Phi(s',a';\Theta'_+)^\top)\Theta_{-1}\| &= r(s,a) \\
\|(\Phi(s,a;\Theta_+)^\top - \gamma\mathbb{E}_{s',a'}\Phi(s',a';\Theta'_+)^\top)\|\|\Theta_{-1}\|\cos\alpha &= r(s,a) \\
\|(\Phi(s,a;\Theta_+)^\top - \gamma\mathbb{E}_{s',a'}\Phi(s',a';\Theta'_+)^\top)\|\|\Theta_{-1}\| &\geq r(s,a) \\
\|(\Phi(s,a;\Theta_+)^\top - \gamma\mathbb{E}_{s',a'}\Phi(s',a';\Theta'_+)^\top)\| &\geq \frac{r(s,a)}{\|\Theta_{-1}\|}
\end{aligned}
\tag{13}
$$

$\square$

In the following, for notational simplicity, we use X_i to denote S_i, A_i for all $i \in [n]$. For any $f \in \mathcal{F}$, $\|f\|_n^2 = 1/n \cdot \sum_{i=1}^n[f(X_i)]^2$. Since both $\hat{O}$ and TQ are bounded by $V_{\max} = R_{\max}/(1-\gamma)$, we only need to consider the case where $\log N_\delta \leq n$.

Let $f_1, \cdots, f_{N_\delta}$ be the centers of minimal δ-covering the of $\mathcal{F}$. By the definition of δ-covering, there exists $k^* \in [N_\delta]$ such that $\|\hat{O} - f_{k^*}\|_\infty \leq \delta$. Notice that k^* is a random variable since $\hat{O}$ is obtained from data.

Theorem 7.2 (One-step Approximation Error of POPRO Update). *Suppose assumption 4.4 hold, let $\mathcal{F} \subseteq \mathcal{B}(\mathcal{S}\times\mathcal{A})$ be a class of measurable function on $\mathcal{S}\times\mathcal{A}$ that are bounded by $V_{\max} = R_{\max}/(1-\gamma)$, and let σ be a probability distribution on $\mathcal{S}\times\mathcal{A}$. Also, let $\{(S_i, A_i)\}_{i\in[n]}$ be n i.i.d. random variables in following σ. Based on $\{(X_i, A_i, Y_i)\}_{i\in[n]}$, we define $\hat{O}$ as the solution to the lease-square with regularization problem,*

$$
\min_{f\in\mathcal{F}}\frac{1}{n}\sum_{i=1}^n[f(S_i,A_i) - Y_i]^2 + \beta\Phi(s,a;\Theta)\mathbb{E}_{s',a'}(s',a';\Theta')
\tag{14}
$$

At the same time, for any $\delta > 0$, let $\mathcal{N}(\delta, \mathcal{F}, \|\cdot\|_\infty)$ be the

$$
\|\hat{O} - TQ\|_\sigma^2 \leq (1+\epsilon)^2 \cdot \omega(\mathcal{F}) + C \cdot V_{\max}^2/(n\cdot\epsilon) + C' \cdot V_{\max} \cdot \delta + 2\beta \cdot G^2,
\tag{15}
$$

where C and C' are two absolute constants and is defined as

$$
\omega(\mathcal{F}) = \sup_{g\in\mathcal{F}}\inf_{f\in\mathcal{F}}\|f - Tg\|_\sigma.
\tag{16}
$$

Proof. **Step (i):** We relate $\mathbb{E}[\|\hat{O} - TQ\|_n^2]$ with its empirical counterpart $\|\hat{O} - TQ\|_n^2$. Since $Y_i = R_i + \gamma\max_{a\in\mathcal{A}}Q(S_{i+1}, a)$ for each $i \in [n]$. By the definition of $\hat{O}$, for any $f \in \mathcal{F}$ we have

$$
\sum_{i=1}^n[Y_i - \hat{O}(X_i)]^2 + \beta\Phi^\top(X_i;\Theta_{\hat{O}})\mathbb{E}\Phi_{X_{i+1}}(X_{i+1};\Theta'_{\hat{O}}) \leq \sum_{i=1}^n[Y_i - f(X_i)]^2 + \beta\Phi^\top(X_i;\Theta_f)\mathbb{E}\Phi_{X_{i+1}}(X_{i+1};\Theta'_f)
\tag{17}
$$

16

For each $i \in [n]$, we define $\xi_i = Y_i - (TQ)(X_i)$. Then eq. (17) can be rewritten as

$$\|\hat{O} - TQ\|_n^2 \leq \|f - TQ\|_n^2 + \frac{1}{n} \sum_{i=1}^{n} \left[2\xi_i[\hat{O}(X_i) - f(X_i)] + \beta \left(\Phi^\top(X_i; \Theta_f) \mathbb{E} \Phi^\top(X_{i+1}; \Theta'_f) - \Phi^\top(X_i; \Theta_{\hat{O}}) \mathbb{E} \Phi(X_{i+1}; \Theta'_{\hat{O}}) \right) \right]$$

(18)

We start by bounding the value of $\left(\Phi^\top(X_i; \Theta_f) \mathbb{E} \Phi(X_{i+1}; \Theta'_f) - \Phi^\top(X_i; \Theta_{\hat{O}}) \mathbb{E} \Phi(X_{i+1}; \Theta'_{\hat{O}}) \right)$. First, by Cauchy-Schwartz Inequality, we have

$$\left| \Phi(X_i; \Theta_f) \mathbb{E} \Phi(X_{i+1}; \Theta'_f) \right| \leq \sqrt{\|\Phi(X_i; \Theta_{f,+})\|^2} \cdot \sqrt{\|\mathbb{E} \Phi(X_{i+1}; \Theta'_{f,+})\|^2} \leq G^2, \qquad (19)$$

where we used assumption 4.4 for the second inequality. Thus, by triangle inequality, we have

$$\left| \Phi^\top(X_i; \Theta_f) \mathbb{E} \Phi(X_{i+1}; \Theta'_f) - \Phi(X_i; \Theta_{\hat{O}}) \mathbb{E} \Phi(X_{i+1}; \Theta'_{\hat{O}}) \right| \leq 2G^2. \qquad (20)$$

And eq. (18) reduces to

$$\|\hat{O} - TQ\|_n^2 \leq \|f - TQ\|_n^2 + \frac{2}{n} \sum_{i=1}^{n} \left[\xi_i[\hat{O}(X_i) - f(X_i)] + \beta G^2 \right] \qquad (21)$$

Then we bound the rest in the right side of eq. (18). Since both f and Q are deterministic, we have $\mathbb{E}(\|f - TQ\|_n^2) = \|f - TQ\|_\sigma^2$. Moreover, since $\mathbb{E}(\xi_i|X_i) = 0$ by definition, we have $\mathbb{E}[\xi_i \cdot g(X_i)] = 0$ for any bounded and measurable function g. Thus it holds that

$$\mathbb{E} \left\{ \sum_{i=1}^{n} \xi_i \cdot [\hat{O}(X_i) - f(X_i)] \right\} = \mathbb{E} \left\{ \sum_{i=1}^{n} \xi_i \cdot [\hat{O} - (TQ)(X_i)] \right\}. \qquad (22)$$

In addition, by triangle inequality and eq. (22) we have

$$\left| \mathbb{E} \left\{ \sum_{i=1}^{n} \xi_i \cdot [\hat{O}(X_i) - (TQ)(X_i)] \right\} \right| \leq \left| \mathbb{E} \left\{ \sum_{i=1}^{n} \xi_i \cdot [\hat{O} - f_{k^*}(X_i)] \right\} \right| + \left| \mathbb{E} \left\{ \sum_{i=1}^{n} \xi_i \cdot [f_{k^*}(X_i) - (TQ)(X_i)] \right\} \right|,$$

(23)

where f_{k*} satisfies $\|f_{k*}\| \leq \delta$. In the following, we upper bound the two terms on the right side of eq. (23) respectively. For the first term, by applying the Cauchy-Schwarz inequality twice, we have

$$\left| \mathbb{E} \left\{ \sum_{i=1}^{n} \xi_i \cdot [\hat{O} - f_{k^*}(X_i)] \right\} \right| \leq \sqrt{n} \cdot \left| \mathbb{E} \left[\left(\sum_{i=1}^{n} \xi_i^2 \right)^{1/2} \cdot \|\hat{O} - f_{k^*}\|_n \right] \right|$$

(24)

$$\leq \sqrt{n} \cdot [\mathbb{E}(\sum_{i=1}^{n} \xi_i^2)]^{1/2} \cdot [\mathbb{E}(\|\hat{O} - f_{k^*}\|_n^2)]^{1/2} \leq n\delta \cdot [\mathbb{E}(\xi_i^2)]^{1/2}.$$

where we use the fact that $\{\xi_i\}_{i \in [n]}$ have the same marginal distributions and $\|\hat{O} - f_{k*}\|_n \leq \delta$. Since both Y_i and TQ are bounded by $V_{\max}$, ξ_i is a bounded random variable by its definition. Thus, there exists a constant $C_\xi > 0$ depending on ξ such that $\mathbb{E}(\xi_i^2) \leq C_\xi^2 \cdot V_{\max}^2$. Then eq. (24) implies

$$\left| \mathbb{E} \left\{ \sum_{i=1}^{n} \xi_i \cdot [\hat{O}(X_i) - f_{k^*}(X_i)] \right\} \right| \leq C_\xi \cdot V_{\max} \cdot n\delta. \qquad (25)$$

It remains to upper bound the second term on the right side of eq. (23). We define N_δ self-normalized random variables

$$Z_j = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} \xi_i \cdot [f_j(X_i) - (TQ)(X_i)] \cdot \|f_j - (TQ)\|_n^{-1} \qquad (26)$$

for all $j \in [N_\delta]$. Here recall that $\{f_j\}_{j \in [N_\delta]}$ are the centers of the minimal δ-covering of $\mathcal{F}$. Then we have

$$\left| \mathbb{E} \left\{ \sum_{i=1}^{n} \xi_i \cdot [f_{k*}(X_i) - (TQ)(X_i)] \right\} \right| = \sqrt{n} \cdot \mathbb{E}[\|f_{k^*} - TQ\|_n \cdot |Z_{k^*}|]$$

$$\leq \sqrt{n} \cdot \mathbb{E} \left\{ [\|\hat{O} - TQ\|_n + \|\hat{O} - f_{k^*}\|_n] \cdot |Z_{k^*}| \right\} \leq \sqrt{n} \cdot \left\{ [\|\hat{O} - TQ\|_n + \delta] \cdot |Z_{k^*}| \right\}$$

(27)

17

where the first inequality follows from triangle inequality and the second follows from the fact that $\leq \delta$ eq. (27), we obtain

$$
\begin{aligned}
\mathbb{E}\left\{[\|\hat{O} - TQ\|_n + \delta] \cdot |Z_{k^*}|\right\} &\leq \left(\mathbb{E}\left\{[\|\hat{O} - TQ\|_n + \delta]^2\right\}\right)^{1/2} \cdot [\mathbb{E}(Z_{k^*}^2)]^{1/2} \\
&\leq \left(\{\mathbb{E}[\|\hat{O} - TQ\|_n^2]\}^{1/2} + \delta\right) \cdot [\mathbb{E}(\max_{j \in [n]} Z_j^2)]^{1/2}
\end{aligned}
\tag{28}
$$

where the last inequality follows from

$$
\mathbb{E}[\|\hat{O} - TQ\|_n] \leq \{\mathbb{E}[\|\hat{O} - TQ\|_n^2]\}^{1/2}.
\tag{29}
$$

Moreover, since ξ_i is centered conditioning on $\{X_i\}$, ξ_i is a sub-Gaussian random variable. Specifically, there exists an absolute constant $H_\xi > 0$ such that $\|\xi_i\|_{\psi_2} \leq H_\xi \cdot V_{\max}$ for each $i \in [n]$. Here the ψ_2-norm of a random variable W is defined as

$$
\|W\|_{\psi_2} = \sup_{p \geq 1} p^{-1/2} [\mathbb{E}(|W|^p)]^{1/p},
\tag{30}
$$

By the definition of Z_j in eq. (26), conditioning on $\{X_i\}_{i \in [n]}$, $\xi_i \cdot [f_j(X_i) - (TQ)(X_i)]$ is a centered and sub-Guassian random variable with

$$
\|\xi_i \cdot [f_j(X_i) - TQ(X_i)]\|_{\psi_2} \leq H_\xi \cdot V_{\max} \cdot |f_j(X_i) - (TQ)(X_i)|
\tag{31}
$$

Moreover, since Z_j is a summation of independent sub-Guassian random variables, by Lemma 5.9 of [61], the ψ_2-norm of Z_j satisfies

$$
\|Z_j\|_{\psi_2} \leq C \cdot H_\xi \cdot V_{\max} \cdot \|f_j - TQ\|_n^{-1} \cdot \left[\frac{1}{n}\sum_{i=1}^n |[f_j(X_i) - (TQ)(X_i)]|^2\right]^{1/2} \leq C \cdot H_\xi \cdot V_{\max},
\tag{32}
$$

where $C > 0$ is an absolute constant. Furthermore, by Lemma 5.14 and 5.15 of [61], Z_j^2 is a sub-exponential random variable, and its moment-generating function is bounded by

$$
\mathbb{E}\left[\exp(t \cdot Z_j^2)\right] \leq \exp(C \cdot t^2 \cdot H_\xi^4 \cdot V_{\max}^4)
\tag{33}
$$

for any t satisfying $C' \cdot |t| \cdot H_\xi^2 \cdot V_{\max}^2 \leq 1$, where C and C' are two positive absolute constants. Moreover, by Jensen's Inequality, we bound the moment-generating function of $\max_{j \in [N_\delta]} Z_j^2$ by

$$
\mathbb{E}\left[\exp(t \cdot \max_{j \in [N_\delta]} Z_j^2)\right] \leq \sum_{j \in [N_\delta]} \mathbb{E}[\exp(t \cdot Z_j^2)]
\tag{34}
$$

Combining eq. (33) and eq. (34), we have

$$
\mathbb{E}(\max_{j \in [N]} Z_j^2) \leq C^2 \cdot H_\xi^2 \cdot V_{\max}^2 \cdot \log N_\delta
\tag{35}
$$

where $C > 0$ is an absolute constant. Hence, plugging eq. (35) into eq. (27) and eq. (28), we upper bound the second term of eq. (22) by

$$
\begin{aligned}
\left|\mathbb{E}\left\{\sum_{i=1}^n \xi_i \cdot [f_{k^*}(X_i) - (TQ)(X_i)]\right\}\right| & \\
&\leq \left(\{\mathbb{E}\|\hat{O} - TQ\|_n^2\}^{1/2} + \delta\right) \cdot C \cdot H_\xi \cdot V_{\max} \cdot \sqrt{n \cdot \log N_\delta}
\end{aligned}
\tag{36}
$$

Finally, combining eq. (21), eq. (25) and eq. (36), we obtain the following inequality

$$
\begin{aligned}
\mathbb{E}[\|\hat{O} - TQ\|_n^2] &\leq \inf_{f \in \mathcal{F}} \mathbb{E}[\|f - TQ\|_n^2] + C_\xi \cdot V_{\max} \cdot \delta \\
&\quad + \left(\{\mathbb{E}\|\hat{O} - (TQ)\|\}^{1/2} + \delta\right) \cdot C \cdot H_\xi \cdot V_{\max} + \sqrt{\log N_\delta / n} + 2 \cdot \beta \cdot G^2 \\
&\leq C \cdot V_{\max} \sqrt{\log N_\delta / n} + \inf_{f \in \mathcal{F}} \mathbb{E}[\|f - TQ\|_n^2] + C' \cdot V_{\max} \delta + 2 \cdot \beta \cdot G^2
\end{aligned}
\tag{37}
$$

18

where C and C' are two constants. Here in the first inequality we take the infimum over $\mathcal{F}$ because eq. (17) holds for any $f \in \mathcal{F}$, and the second inequality holds because $\log N_\delta \leq n$.

Now we invoke a fact to obtain the final bound for $\mathbb{E}[\|\hat{O} - TQ\|_n^2]$ from eq. (37). Let a, b and c be positive numbers satisfying $a^2 \leq 2ab + c$. For any $\epsilon \in (0, 1]$, since $2ab \leq \frac{\epsilon}{1+\epsilon} a^2 + \frac{1+\epsilon}{\epsilon} b^2$, we have

$$a^2 \leq (1 + \epsilon)^2 \cdot b^2/\epsilon + (1 + \epsilon) \cdot c \tag{38}$$

Therefore, applying eq. (38) to eq. (37) with $a^2 = \mathbb{E}[\|\hat{O} - TQ\|_n^2]$, $b = C \cdot V_{\max} \cdot \sqrt{\log N}$ and $c = \inf_{f \in \mathcal{F}} \mathbb{E}[\|f - TQ\|_n^2] + C' \cdot V_{\max} \cdot \delta$, we obtain

$$\mathbb{E}[\|\hat{O} - TQ\|_n^2] \leq (1 + \epsilon) \cdot \inf_{f \in \mathcal{F}} \mathbb{E}[\|f - TQ\|_n^2] + C \cdot V_{\max}^2 \cdot \log N_\delta/(n\epsilon) + C' \cdot V_{\max} \cdot \delta + 2\beta G^2, \tag{39}$$

where C and C' are two positive absolute constants. This concludes the first step.

Step (ii): In this step, we relate the population risk $\|\hat{O} - TQ\|_\delta^2$ with $\mathbb{E}[\|\hat{O} - TQ\|_n^2]$, which is bounded in the first step. To begin with, we generate n i.i.d. random variables $\{\tilde{X}_i = (\tilde{S}_i, \tilde{A}_i)\}_{i \in [n]}$ following σ, independent of $\{(S_i, A_i, R_i, S_i')\}_{i \in [n]}$. Since $\|\hat{O} - f_{k^*}\|_\infty \leq \delta$, for any $x \in \mathcal{S} \times \mathcal{A}$, we have

$$\left|[\hat{O}(x) - (TQ)(x)]^2 - [f_{k^*}(x) - (TQ)(x)]^2\right|$$
$$= \left|\hat{O}(x) - f_{k^*}(x)\right| \cdot \left|\hat{O}(x) + f_{k^*}(x) - 2(TQ)(x)\right| \leq 4V_{\max} \cdot \delta, \tag{40}$$

where the last inequality follows from the fact that $\|TQ\|_\infty \leq V_{\max}$ and $\|f\|_\infty \leq V_{\max}$ for any $f \in \mathcal{F}$.

Then by the definition of $\|\hat{O} - TQ\|_\delta^2$ and eq. (40), we have

$$\|\hat{O} - TQ\|_\sigma^2 = \mathbb{E}\left\{\frac{1}{n} \sum_{i=1}^n [\hat{O}(\tilde{X}_i) - (TQ)(\tilde{X}_i)]^2\right\}$$
$$\leq \mathbb{E}\left\{\|\hat{O} - TQ\|_n^2 + \frac{1}{n} \sum_{i=1}^n [f_{k^*}(\tilde{X}_i) - (TQ)(\tilde{X}_i)]^2 - \frac{1}{n} \sum_{i=1}^n [f_{k^*}(X_i) - (TQ)(\tilde{X}_i)]^2\right\} + 8V_{\max} \cdot \delta$$
$$= \mathbb{E}(\|\hat{O} - TQ\|_n^2) + \mathbb{E}[\frac{1}{n} \sum^n h_{k^*}(X_i, \tilde{X}_i)] + 8V_{\max} \cdot \delta \tag{41}$$

where we apply eq. (40) to obtain the first inequality, and in the last equality we define

$$h_j(x, y) = [f_j(y) - (TQ)(y)]^2 - [f_j(x) - (TQ)(x)]^2, \tag{42}$$

for any $x, y \in \mathcal{S} \times \mathcal{A}$ and any $j \in [N_\delta]$. Note that h_{k^*} is a random function since k^* is random. By the definition of h_j, we have $|h_j(x, y)| \leq 4V_{\max}^2$ for any $(x, y) \in \mathcal{S} \times \mathcal{A}$ and $\mathbb{E}[h_j(X_i, \tilde{X}_i)] = 0$ for any $i \in [n]$. Moreover, the variance of $h_j(X_i, \tilde{X}_i)$ satisfies

$$\text{Var}[h_j(X_i, \tilde{X}_i)] = 2 \text{Var}\left\{[f_j(X_i) - (TQ)(X_i)]^2\right\}$$
$$\leq 2\mathbb{E}\left\{[f_j(X_i) - (TQ)(X_i)]^4\right\} \leq 8\Upsilon^2 \cdot V_{\max}^2, \tag{43}$$

where we define Υ by letting

$$\Upsilon = \max(4V_{\max}^2 \cdot \log N_\delta/n, \max_{j \in [N_\delta]} \mathbb{E}\left\{[f_j(X_i) - (TQ)(X_i)]^2\right\}) \tag{44}$$

Furthermore, we define

$$T = \sup_{j \in [N_\delta]} \left|\sum_{i=1}^n h(X_i, \tilde{X}_i)/\Upsilon\right| \tag{45}$$

Combining eq. (41) and eq. (45),

$$\|\hat{O} - TQ\|_\sigma^2 \leq \mathbb{E}[\|\hat{O} - TQ\|_n^2] + \Upsilon/n \cdot \mathbb{E}[T] + 8V_{\max} \cdot \delta. \tag{46}$$

In the following, we use Bernstein's Inequality to establish an upper bound for $\mathbb{E}(T)$:

19

Lemma 7.3. *(Bernstein's Inequality) Let $U_1, \cdots, U_n$ be n independent random variables satisfying $\mathbb{E}(U_i) = 0$ and $\leq$ for all $i \in [n]$. Then for any $t > 0$, we have*

$$\mathbb{P}\left(\left|\sum_{i=1}^{n} U_i\right| \geq t\right) \leq 2\exp\left(\frac{-t^2}{2M \cdot t/3 + 2\sigma^2}\right) \tag{47}$$

where $\sigma^2 = \sum_{i=1}^{n}$ is the variance of $\sum_{i=1}^{n} U_i$.

We first apply Bernstein's Inequality by setting $U_i = h_j(X_i, \tilde{X}_i)/\Upsilon$ for each $i \in [n]$. Then we take a union bound for all $j \in [N_\delta]$ to obtain

$$\mathbb{P}(T \geq t) = \mathbb{P}\left[\sup_{j \in [N_\delta]} \frac{1}{n}\left|\sum_{i=1}^{n} h_j(X_i, \tilde{X}_i)/\Upsilon\right| \geq t\right] \leq 2N_\delta \cdot \exp\left\{\frac{-t^2}{8V_{\max}^2 \cdot [t/(3\Upsilon) + n]}\right\} \tag{48}$$

Since T is nonnegative, $\mathbb{E}(T) = \int_0^\infty \mathbb{P}(T \geq t)dt$. Thus, for any $u \in (0, 3\Upsilon \cdot n)$,

$$\mathbb{E}(T) \leq u + \int_u^\infty \mathbb{P}(T \geq t)\, dt \leq u + 2N_\delta \int_u^{3\Upsilon \cdot n} \exp\left(\frac{-t^2}{16V_{\max}^2 \cdot n}\right)dt + 2N_\delta \int_{3\Upsilon \cdot n}^\infty \exp\left(\frac{-3\Upsilon \cdot t}{16V_{\max}^2}\right)dt$$

$$\leq u + 32N_\delta \cdot V_{\max} \cdot n/u \cdot \exp\left(\frac{-u^2}{16V_{\max}^2 \cdot n}\right) + 32N_\delta \cdot V_{\max}^2/(3\Upsilon) \cdot \exp\left(\frac{-9\Upsilon^2 \cdot n}{16V_{\max}^2}\right) \tag{49}$$

where in the second inequality we use the fact that $\int_s^\infty \exp(-t^2/2)dt \leq 1/s \cdot \exp(-s^2/2)$. Now we set $u = 4V_{\max}\sqrt{n \cdot \log N_\delta}$ in eq. (49) and plug in the definition of Υ in eq. (43) to obtain

$$\mathbb{E} \leq 4V_{\max}\log n \cdot N_\delta + 8V_{\max}\sqrt{n/\log N_\delta} + 6V_{\max}\sqrt{n/\log N_\delta} \leq 8V_{\max}\sqrt{n \cdot \log N_\delta}, \tag{50}$$

where the last inequality holds when $\log N_\delta \geq 4$. Moreover, the definition of Υ in eq. (43) implies that $\Upsilon \leq \max[2V_{\max}\sqrt{\log N_\delta/n}, \|\hat{O} - TQ\|_\sigma^2 + \delta]$. In the following, we only need to consider the case where $\Upsilon \leq \|\hat{O} - TQ\|_\sigma + \delta$, since we already have eq. (15) if $\|\hat{O} - TQ\| + \delta \leq 2V_{\max}\sqrt{\log N_\delta/n}$, which concludes the proof.

Then, when $\Upsilon \leq |\hat{O} - TQ\|_\sigma + \delta$ holds, combining eq. (46) and eq. (50) we have,

$$\|\hat{O} - TQ\|_\delta^2 \leq \mathbb{E}[\|\hat{O} - TQ\|_n^2] + 8V_{\max}\sqrt{\log N_\delta/n} \cdot \|\hat{O} - TQ\|_\delta + 8V_{\max}\sqrt{\log N_\delta/n} \cdot \delta + 8V_{\max} \cdot \delta$$

$$\leq \mathbb{E}[\|\hat{O} - TQ\|_n^2] + 8V_{\max}\sqrt{\log N_\delta/n} \cdot \|\hat{O} - TQ\|_\sigma + 16V_{\max} \cdot \delta \tag{51}$$

We apply the inequality in eq. (38) to eq. (51) with $a = \|\hat{O} - TQ\|_\sigma$, $b = 8V_{\max}\sqrt{\log N_\delta/n}$, and $c = \mathbb{E}[\|\hat{O} - TQ\|_n^2] + 16V_{\max} \cdot \delta$ we have. Hence we found

$$\|\hat{O} - TQ\|_\sigma^2 \leq (1 + \epsilon) \cdot \mathbb{E}[\|\hat{O} - TQ\|_n^2] + (1 + \epsilon)^2 \cdot 64V_{\max} \cdot \log N_\delta/(n \cdot \epsilon) + (1 + \epsilon) \cdot 18V_{\max} \cdot \delta \tag{52}$$

which concludes the second step of the proof.

Finally, combining steps **(i)** and together, i.e., eq. (39) and eq. (52), we conclude that

$$\|\hat{O} - TQ\|_\sigma^2 \leq (1 + \epsilon)^2 \cdot \inf_{f \in \mathcal{F}} \mathbb{E}[\|f - TQ\|_n^2] + C_1 \cdot V_{\max}^2 \cdot \log N_\delta/(n \cdot \epsilon) + C_2 \cdot V_{\max} \cdot \delta + 2\beta G^2, \tag{53}$$

where C_1 and C_2 are two absolute constants. Moreover, since $Q \in \mathcal{F}$

$$\inf_{f \in \mathcal{F}} \mathbb{E}[\|f - TQ\|_n^2] \leq \sup_{Q \in \mathcal{F}}\left\{\inf_{f \in \mathcal{F}} \mathbb{E}[\|f - TQ\|_n^2]\right\} \tag{54}$$

which concludes the proof of theorem 4.5. $\qquad\square$

8   Appendix B: Experimental Settings

In this section, we provide the experimental settings in detail.

8.1   Anonymous code

Our anonymous code can be found at `https://anonymous.4open.science/r/POPRO-7A4E/README.md`.

8.2   Training details

Computational resources. All experiments are conducted on two GPU servers. The first one has 3 Titan XP GPUs and Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz. The second one has 4 Titan RTX GPUs and an Intel(R) Xeon(R) Gold 6137 CPU @ 3.90GHz. Each random seed for DMControl takes 2 days to finish. For PyBullet and MuJoCo tasks, it takes 5 hours to finish a random seed. For PyBullet and MuJoCo suite, we simultaneously launch 70 seeds. For the DMControl suite, we simultaneously run 15 random seeds.

Random seeds. If not otherwise specified, we evaluate each tested algorithm over 10 random seeds to ensure the reproducibility of our experiments. Also, we set all seeds fixed in our experiments.

PyBullet. When we train the agent in the Pybullet suite, the agent starts by randomly collecting 25,000 states and actions for better exploration. Then we evaluate the agent for ten episodes every 5,000 timesteps. We take the average return of ten episodes as a key evaluation metric. Note that for a more fair evaluation of the algorithms, at the evaluation phase, we do not apply any exploration tricks in the algorithms (e.g. injecting noise into actions in TD3), because these exploration tricks may harm the performance of tested algorithms. The complete timesteps are 1 million. The results are reported over ten random seeds. For the hyper-parameter β of POPRO, we take $5e-4$ for every task and never change it. Note that statistics in Table table 2 is slightly different from fig. 3 due to the figure taking windows smoothing for more clear visual effect.

Except for METD3 [24], we use the author's implementation [15] or commonly used public repository [59]. Our implementations of POPRO and METD3 are based on TD3 implementation. To fairly evaluate our algorithm, we keep all the original TD3's hyper-parameters without any modification. For the hyper-parameter of METD3, we set dropout rate equal to 0.1 as the author [24] did. The soft update style is adopted for METD3, POPRO with $\eta = 0.005$. We summarize the hyper-parameter settings in table 3.

MuJoCo. All experiments on mujoco are consistent with the PyBullet settings, except for the code of SAC used. We find that the performance of SAC [58] collapse in the MuJoCo suite, thus we adopt the code of Stable-Baselines3 [1] [62] for SAC implementation with the same hyper-parameters under PyBullet settings.

DMControl. We utilize the authors' implementation of CURL without any modification as we discussed. And we do not change the default hyper-parameters for SAC, CURL[2]. For a fair comparison, we keep the hyper-parameters of CURL the same as CURL. And the hyper-parameter $\beta = 5e-4$ is kept in each environment. We summarize the hyper-parameter settings for the DMControl suite in table 4.

We present the experimental results of CURL [5] and DrQ [9] in table 1 as the authors' reports.

8.3   Missing Algorithms

Our POPRO implementations are based on TD3 and CURL respectively. We present POPRO based on TD3 in algorithm 2.

---

[1]Code: `https://github.com/DLR-RM/stable-baselines3`
[2]Code: `https://github.com/MishaLaskin/curl`

Table 3: Hyper-parameters settings for PyBullet experiments

Hyper-parameter	Value
Shared hyper-parameters	
discount (γ)	0.99
Replay buffer size	10^6
Optimizer	Adam [63]
Learning rate for actor	3×10^{-4}
Learning rate for critic	3×10^{-4}
Number of hidden layer for all networks	2
Number of hidden units per layer	256
Activation function	ReLU
Mini-batch size	256
Random starting exploration time steps	2.5×10^4
Target smoothing coefficient (η)	0.005
Gradient Clipping	False
Target update interval (d)	2
TD3	
Variance of exploration noise	0.2
Variance of target policy smoothing	0.2
Noise clip range	$[-0.5, 0.5]$
Delayed policy update frequency	2
POPRO	
pro coefficient (β)	5×10^{-4}
SAC	
Target Entropy	- dim of $\mathcal{A}$
Learning rate for α	1×10^{-4}

Algorithm 2 POPRO (based on TD3)

1: Initialize actor network π, and critic network Q_i for $i = 1, 2$ with random parameters ϕ, θ_1, θ_2
2: Initialize target networks $\theta_i' \leftarrow \theta_i$, $\phi' \leftarrow \phi$
3: Initialize replay buffer $\mathcal{B}$
4: Initialize $\beta, d, \sigma, \tilde{\sigma}, \eta, c$ total steps T, and $t = 0$
5: Reset the environment and receive initial state s
6: **while** $t < T$ **do**
7:     Select action with noise $a = \pi(s; \phi) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2)$, and receive reward r, new state s'
8:     Store transition tuple (s, a, r, s') in $\mathcal{B}$
9:     Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
10:     $\tilde{a} \leftarrow \pi(s'; \phi') + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}^2), -c, c)$
11:     $y \leftarrow r + \gamma \min_{i=1,2} Q(s', \tilde{a}; \theta_i')$
12:     Update critic by minimizing eq. (8)
13:     $\theta \leftarrow \arg\min_{\theta_i} N^{-1} \sum (y_i - Q(s_i, a_i; \theta))^2 + \beta \Phi(s, a; \theta_+)^\top N^{-1} \sum \left[\Phi(s_i', \tilde{a}_i; \theta_+') \right]$
14:     **if** t mod d **then**
15:         Update ϕ by DPG [49]:
16:         $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q(s, a; \theta_1)|_{a=\pi(s; \phi)} \nabla_\phi \pi(s; \phi)$
17:         Update target networks:
18:         $\theta_i' \leftarrow \eta \theta_i + (1 - \eta) \theta_i'$
19:         $\phi' \leftarrow \eta \phi + (1 - \eta) \phi'$
20:     **end if**
21:     $t \leftarrow t + 1$
22:     $s \leftarrow s'$
23: **end while**

Table 4: Hyper-parameters settings for DMControl experiments

Hyper-parameter	Value
Shared hyper-parameters	
Discount γ	0.99
Replay buffer size	100000
Optimizer	Adam
Learning rate	1×10^{-4}
Learning rate $(f_\theta, \pi_\psi, Q_\phi)$	2×10^{-4} cheetah, run
	1×10^{-3} otherwise
Convolutional layers	4
Number of filters	32
Activation function	ReLU
Encoder EMA η	0.05
Q function EMA (η)	0.01
Mini-batch size	512
Target Update interval (d)	2
Latent dimension	50
Initial temperature	0.99
Number of hidden units per layer (MLP)	1024
Evaluation episodes	10
Random crop	True
Observation rendering	(100,100)
Observation downsampling	(84,84)
Initial steps	1000
Stacked frames	3
Action repeat	2 finger, spin; walker, walk
	8 cartpole, swingup
	4 otherwise
$(\beta_1, \beta_2) \rightarrow (f_\theta, \pi_\psi, Q_\phi)$	(.9, .999)
$(\beta_1, \beta_2) \rightarrow (\alpha)$	(.9, .999)
POPRO	
pro coefficient (β)	5×10^{-4}

9 Appendix C: Additional Experimental Results

In this section, we provide additional experimental results.

9.1 Similarity measures

We present experiments under Manhattan distance and cosine similarity measures in fig. 5.
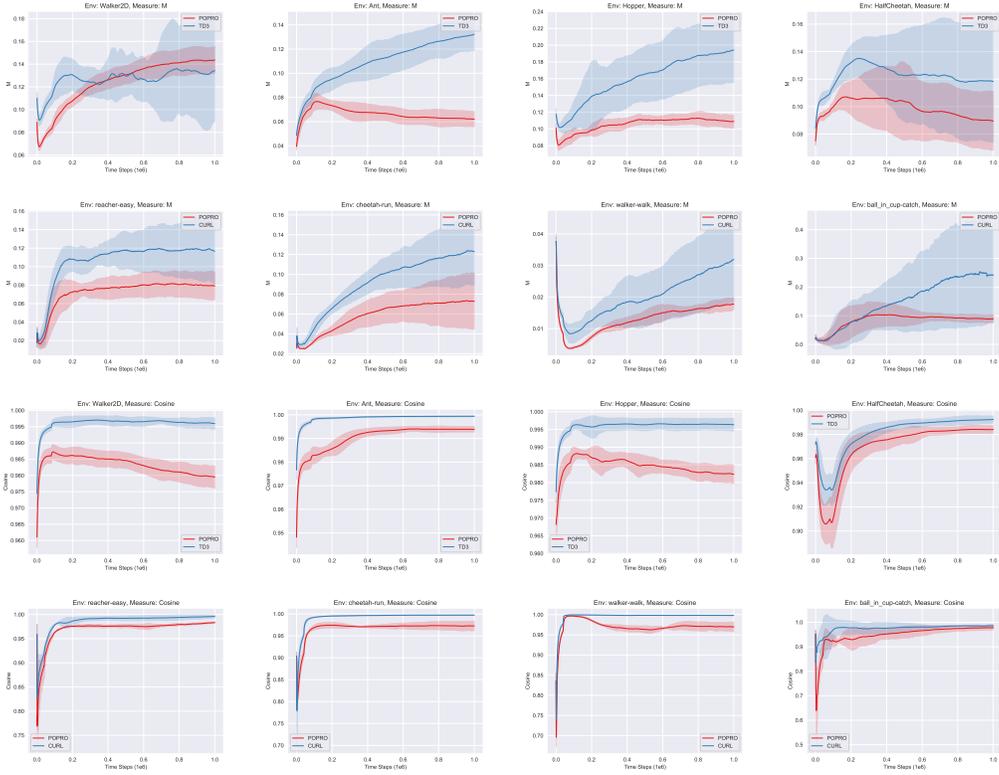

Figure 5: Similarity measures for representation of action value functions of TD3 and POPRO agents. The shaded area stands for a standard deviation. Column: various environments. Row: different algorithm. The representations of action value networks of TD3 and CURL agents grow similar as training processing, which results in the collapse of representation gap. But POPRO framework does not.

9.2 Experiments on MuJoCo suite

We present the performance of POPRO on MuJoCo suite in table 5. The results show that our proposed framework POPRO outperforms or matches the compared algorithms in 5 out 7 MuJoCo environments. Compared with its backbone algorithm TD3, the POPRO framework outperforms it in 6 out of 7 environments.

9.3 Experiments on 16 DMControl tasks

We present the performance curves of POPRO on a total on a total of 16 DMControl environments in fig. 6. We run 4 seeds in each environments. The POPRO framework always achieves good performance in the environments tested.

Table 5: The average return of the last ten evaluations over ten random seeds. The maximum average returns are bolded. POPRO outperforms or matches the other tested algorithms in 5 out of 7 environments.

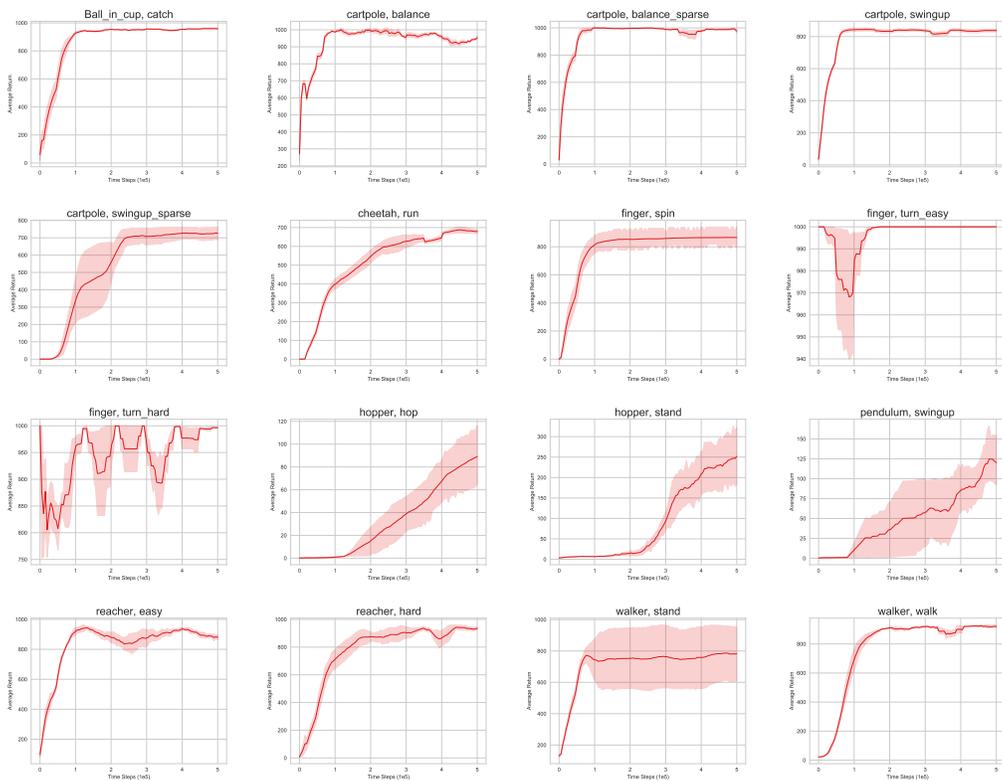| Algorithm	Ant	HalfCheetah	Hopper	InvDouPen	InvPen	Reacher	Walker
POPRO	**5386 ± 493**	10832 ± 501	**3424 ± 180**	7470 ± 3721	**1000 ± 0**	**- 4 ± 1**	**4223 ± 655**
TD3	5102 ± 787	**10858 ± 637**	3163 ± 367	7312 ± 3653	1000 ± 0	- 4 ± 1	3762 ± 956
METD3	2256 ± 431	5696 ± 1740	804 ± 71	7815 ± 0	912 ± 71	- 8 ± 3	2079 ± 1096
SAC	4233 ± 806	10482 ± 959	2666 ± 320	**9358 ± 0**	1000 ± 0	- 4 ± 0	4187 ± 304

Figure 6: Performance curves for DMControl suite. The shaded region represents a standard deviation of the average evaluation over 4 seeds. The curves are smoothed by moving average. The POPRO framework always achieves good performance in the environments tested.