Frustratingly Easy Regularization on Representation Can Boost Deep Reinforcement Learning

Qiang He[1]   Huangyuan Su[2]   Jieyu Zhang[3]   Xinwen Hou[1]

[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]Carnegie Mellon University, Pittsburgh, United States
[3]University of Washington, Seattle, United States

qianghe97@gmail.com, huangyus@andrew.cmu.edu, jieyuz2@cs.washington.edu, xinwen.hou@ia.ac.cn

Abstract

Deep reinforcement learning (DRL) 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. In this work, we demonstrate that the learned representation of the Q-network and its target Q-network should, in theory, satisfy a favorable distinguishable representation property. Specifically, there exists an upper bound on the representation similarity of the value functions of two adjacent time steps in a typical DRL setting. However, through illustrative experiments, we show that the learned DRL agent may violate this property and lead to a suboptimal policy. Therefore, we propose a simple yet effective regularizer called Policy Evaluation with Easy Regularization on Representation (PEER), which aims to maintain the distinguishable representation property via explicit regularization on internal representations. And we provide the convergence rate guarantee of PEER. Implementing PEER requires only one line of code. Our experiments demonstrate that incorporating PEER into DRL can significantly improve performance and sample efficiency. Comprehensive experiments show that PEER achieves state-of-the-art performance on all 4 environments on PyBullet, 9 out of 12 tasks on DM-Control, and 19 out of 26 games on Atari. To the best of our knowledge, PEER is the first work to study the inherent representation property of Q-network and its target. Our code is available at https://sites.google.com/view/peer-cvpr2023/.

1. Introduction

Deep reinforcement learning (DRL) leverages the function approximation abilities of deep neural networks (DNN) and the credit assignment capabilities of RL to enable agents to perform complex control tasks using high-dimensional observations such as image pixels and sensor information [1, 2, 3, 4]. DNNs are used to parameterize the policy and value functions, but this requires the removal of irrelevant and redundant information while retaining pertinent information, which is the task of representation learning. As a result, representation learning has been the focus of attention for researchers in the field [5, 6, 7, 8, 9, 10, 11, 12, 13]. In this paper, we investigate the inherent representation properties of DRL.

The action-value function is a measure of the quality of taking an action in a given state. In DRL, this function is approximated by the action-value network or Q-network. To enhance the training stability of the DRL agent, Mnih et al. [2] introduced a target network, which computes the target value with the frozen network parameters. The weights of a target network are either periodically replicated from learning Q-network, or exponentially averaged over time steps. Despite the crucial role played by the target network in DRL, previous studies[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24] have not considered the representation property of the target network. In this work, we investigate the inherent representation property of the Q-network. Following the commonly used definition of representation of Q-network [17, 25, 26], the Q-network can be separated into a nonlinear encoder and a linear layer, with the representation being the output of the nonlinear encoder. By employing this decomposition, we reformulate the Bellman equation [27] from the perspective of representation of Q-network and its target. We then analyze this formulation and demonstrate theoretically that a favorable *distinguishable representation property* exists between the representation of Q-network and that of its target. Specifically, there exists an upper bound on the representation similarity of the value functions of two adjacent time steps in a typical DRL setting, which differs from previous work.

We subsequently conduct experimental verification to investigate whether agents can maintain the favorable dis-

tinguishable representation property. To this end, we choose two prominent DRL algorithms, TD3 [28] and CURL [6] (without/with explicit representation learning techniques). The experimental results indicate that the TD3 agent indeed maintains the distinguishable representation property, which is a positive sign for its performance. However, the CURL agent fails to preserve this property, which can potentially have negative effects on the model's overall performance.

These theoretical and experimental findings motivate us to propose a simple yet effective regularizer, named Policy Evaluation with Easy Regularization on Representation (PEER). PEER aims to ensure that the agent maintains the distinguishable representation property via explicit regularization on the Q-network's internal representations. Specifically, PEER regularizes the policy evaluation phase by pushing the representation of the Q-network away from its target. Implementing PEER requires only one line of code. Additionally, we provide a theoretical guarantee for the convergence of PEER.

We evaluate the effectiveness of PEER by combining it with three representative DRL methods TD3 [28], CURL [6], and DrQ [9]. The experiments show that PEER effectively maintains the distinguishable representation property in both state-based PyBullet [29] and pixel-based DMControl [30] suites. Additionally, comprehensive experiments demonstrate that PEER outperforms compared algorithms on four tested suites PyBullet, MuJoCo [31], DMControl, and Atari [32]. Specifically, PEER achieves state-of-the-art performance on **4** out of **4** environments on PyBullet, **9** out of **12** tasks on DMControl, and **19** out of **26** games on Atari. Moreover, our results also reveal that combining algorithms (e.g., TD3, CURL, DrQ) with the PEER loss outperforms their respective backbone methods. This observation suggests that the performance of DRL algorithms may be negatively impacted if the favorable distinguishable representation property is not maintained. The results also demonstrate that the PEER loss is orthogonal to existing representation learning methods in DRL.

Our contributions are summarized as follows. (i) We theoretically demonstrate the existence of a favorable property, *distinguishable representation property*, between the representation of Q-network and its target. (ii) The experiments show that learned DRL agents may violate such a property, possibly leading to sub-optimal policy. To address this issue, we propose an easy-to-implement and effective regularizer PEER that ensures that the property is maintained. To the best of our knowledge, the PEER loss is the first work to study the inherent representation property of Q-network and its target and be leveraged to boost DRL. (iii) In addition, we provide the convergence rate guarantee of PEER. (iv) To demonstrate the effectiveness of PEER, we perform comprehensive experiments on four commonly used RL suits PyBullet, MuJoCo, DMControl, and Atari suites. The empir-

ical results show that PEER can dramatically boost state-of-the-art representation learning DRL methods.

2. Preliminaries

DRL aims to optimize the policy through return, which is defined as $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$.

Q-network and its target. Action value function $Q^\pi(s, a)$ represents the quality of a specific action a in a state s. Formally, the action value (Q) function is defined as

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

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 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 [27]

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

where $s' \sim p(\cdot|s, a)$ and $a' \sim \pi(s')$. The process of evaluating value function is known as the policy evaluation phase. To stabilize the training of DRL, Mnih et al. [2] introduced a target network to update the learning network with $\theta' \leftarrow \eta\theta + (1 - \eta)\theta'$, where η is a small constant controlling the update scale. θ is the parameters of the Q-network. And θ' denotes the parameters of the target network.

Representation of Q-network. We consider a multi-layer neural network representing the Q function parameterized by Θ. Let Θ_i represent the parameters of $i-$th layer, Θ_{-1} represent the parameters of the last layer, and Θ_+ represent the parameters of the neural networks except for those of the last layer. The representation Φ of the Q-network is defined as

$$Q(s, a; \Theta) = \langle \Phi(s, a; \Theta_+), \Theta_{-1} \rangle. \qquad (3)$$

One intuitive way to comprehend the representation of the Q-network is to view the network as composed of a non-linear encoder and a linear layer. The representation of the Q-network is the output of the encoder [17, 25, 26, 33, 34].

3. Method

In this section, we start with a theoretical analysis of the Q-network and demonstrate the existence of the distinguishable representation property. The preliminary experiments (fig. 1) reveal the desirable property may be violated by agents learned with existing methods. To ensure agents maintain such property and therefore alleviate the potential negative impact on model performance, we propose Policy Evaluation with Easy Regularization on Representation (PEER), a simple yet effective regularization loss on the representation of the Q-/target network. Finally, we employ a toy example to demonstrate the efficacy of the proposed PEER. We defer all the proofs to Appendix section 7.
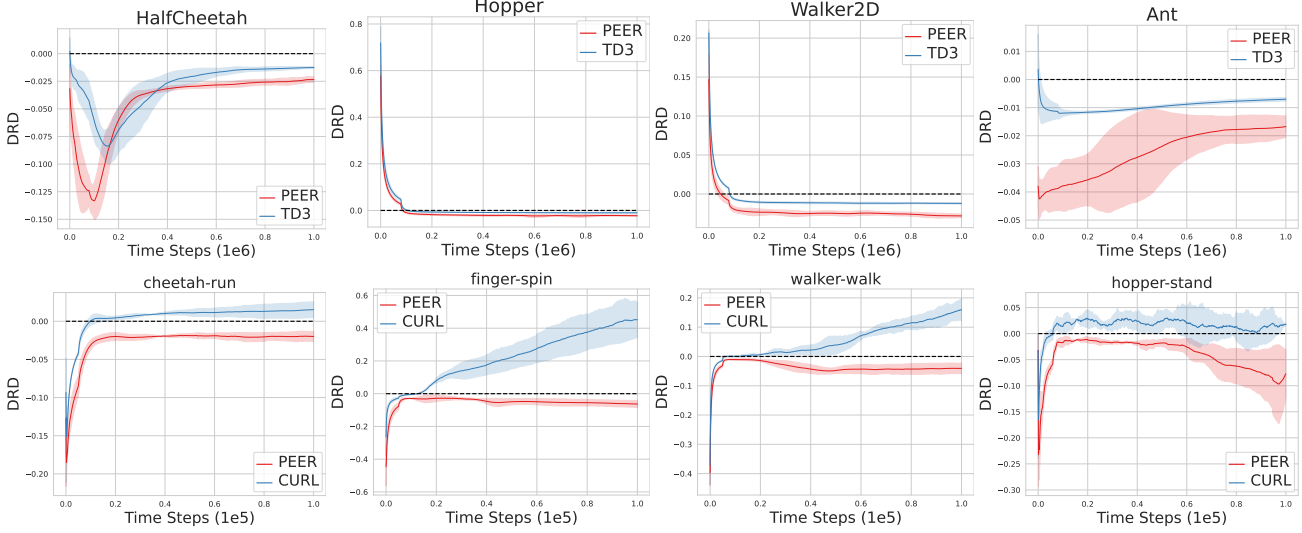
Figure 1. Distinguishable representation discrepancy (DRD) of TD3, CURL, and PEER agents on PyBullet and DMControl suites. TD3 agent does enjoy the distinguishable representation property. But the CURL agent does not satisfy the distinguishable representation property, which might negatively affect the model performance (see table 2). Not only does PEER enjoy the distinguishable representation property on state-based inputs environment bullets, but also in the pixel-based environment DMControl. The shaded area stands for a standard deviation.

3.1. Theoretical analysis

First of all, we theoretically examine the property that the internal representations of a Q-network and its target should satisfy under assumption 3.1. Specifically, we show that the similarity of internal representations among two adjacent action-state pairs must be below a constant value, determined by both the neural network and reward function.

Assumption 3.1. The l_2-norm of the representation is uniformly bounded by the square of some positive constant G, i.e. $\|\Phi(X; \Theta_+)\|^2 \leq G^2$ for any $X \in \mathcal{S} \times \mathcal{A}$ and network weights Θ.

Theorem 3.2 (Distinguishable Representation Property). *The similarity (defined as inner product $\langle \cdot, \cdot \rangle$) between normalized representations $\Phi(s, a; \Theta_+)$ of the Q-network and $\mathbb{E}_{s',a'}\Phi(s', a'; \Theta'_+)$ satisfies*

$$\langle \Phi(s, a; \Theta_+), \mathbb{E}_{s',a'}\Phi(s', a'; \Theta'_+)\rangle \leq \frac{1}{\gamma} - \frac{r(s, a)^2}{2\|\Theta_{-1}\|^2}, \quad (4)$$

where s, a and Θ_+ are state, action, and parameters of the Q-network except for those of the last layer. While s', a', Θ'_+ are the state, action at the next time step, and parameters of the target Q-network except for those of the last layer. And Θ_{-1} is the parameters of the last layer of Q-network.

Intuitively, theorem 3.2 indicates that the internal representations of different action-state pairs should be *distinguishable* so that the model is able to better pick the right action from action space. We define distinguishable repre-

sentation discrepancy (DRD) as

$$\text{DRD} = \langle \Phi(s, a; \Theta_+), \mathbb{E}_{s',a'}\Phi(s', a'; \Theta'_+)\rangle - \left(\frac{1}{\gamma} - \frac{r(s, a)^2}{2\|\Theta_{-1}\|^2}\right). \quad (5)$$

We determine whether the internal representations of the Q-network and its target satisfy the distinguishable representation property by examining the value of DRD. If DRD ≤ 0, then the property is satisfied; otherwise, it is not. Then, we evaluate whether this property is maintained by agents learned with two representative DRL methods, namely TD3 [28] and CURL [6] (without/with explicit representation learning techniques). The results are shown in fig. 1. From the results, we see that the TD3 agent does preserve the distinguishable representation property. While the CURL agent using explicit representation learning techniques does not satisfy the distinguishable representation property. This violation of the property could have a detrimental impact on agent performance (see section 4.2). These theoretical and experimental results motivate us to propose the following PEER regularization loss. This loss ensures that the learned model satisfies the distinguishable representation property through explicit regularization on representations.

3.2. PEER

Specifically, the PEER loss is defined as

$$\mathcal{L}_{\text{PEER}}(\Theta) = \langle \Phi(s, a; \Theta_+), \mathbb{E}_{s',a'}\left[\Phi(s', a'; \Theta'_+)\right]\rangle, \quad (6)$$

where the Q-network takes the current state and action as inputs, and the inputs to the target network are the state and action at the next time step. This simple loss can be readily

$$\mathcal{L} = \mathcal{L}_{\text{PE}} + \underbrace{\langle \Phi(s, a; \Theta_+), \mathbb{E}_{s', a'} \Phi(s', a'; \Theta'_+) \rangle}_{\mathcal{L}_{\text{PEER}}}$$
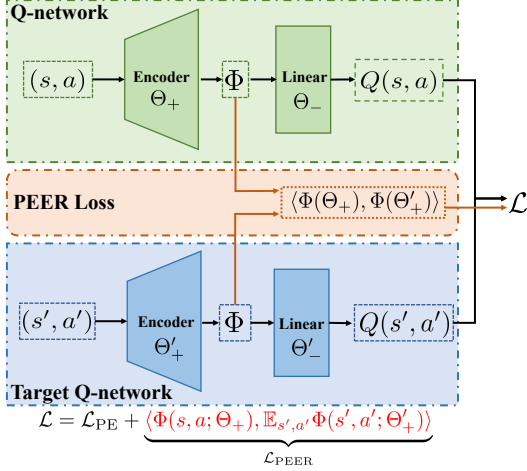
Figure 2. How the PEER loss is computed. The encoder is a nonlinear operator, and the state action pairs generate the representation Φ through the encoder and then the action value through a linear layer. PEER regularizes the policy evaluation phase by differing the representation Φ of the Q-network from that of its target. $\mathcal{L}_{\text{PE}}$ is a form of policy evaluation loss. And β is a small positive constant, controlling the magnitude of the regularization effectiveness.

incorporated with the optimization objective of standard policy evaluation methods [2, 4, 35], leading to

$$\mathcal{L}(\Theta) = \mathcal{L}_{\text{PE}}(\Theta) + \beta \mathcal{L}_{\text{PEER}}(\Theta), \qquad (7)$$

where hyper-parameter β controls the magnitude of the regularization effect of PEER. And $\mathcal{L}_{\text{PE}}(\Theta)$ is a policy evaluation phase loss e.g.

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

PEER can be combined with any DRL method that includes a policy evaluation phase, such as DQN [2], TD3, SAC [4]. Experiments presented in fig. 1 demonstrate that PEER maintains the distinguishable representation property. Furthermore, extensive experiments demonstrate PEER significantly improves existing algorithms by keeping the such property. The Pytorch-like pseudocode for the PEER loss can be found in Appendix section 8.3. And fig. 2 illustrates how the PEER loss is computed.

Convergence Guarantee. We additionally provide a convergence guarantee of our algorithm. Following the definition in [36], let $\mathcal{F}$ be a class of measurable functions, a δ-cover for $\mathcal{F}$ with $\delta > 0$ is a finite set $\Gamma_\delta \subset \mathcal{F}$ such that $\forall f \in \mathcal{F}$, there exists $g \in \Gamma_\delta$ such that $\|f - g\|_\infty \le \delta$, where $\|\cdot\|_\infty$ is the l_∞-norm. A minimal δ-cover is a δ-cover and if taking out any of its elements, it is no longer a δ-cover for $\mathcal{F}$. Let T be the Bellman Operator. We have the following convergence result for the core update step in PEER.

Theorem 3.3 (One-step Approximation Error of the PEER Update). *Suppose assumption 3.1 hold, let $\mathcal{F} \subset \mathcal{B}(\mathcal{S} \times \mathcal{A})$ 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 $\mathcal{S} \times \mathcal{A}$ 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,*

$$\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n [f(S_i, A_i) - Y_i]^2 + \beta \langle \Phi(s, a; \Theta), \mathbb{E}\Phi_{s', a'}(s', a'; \Theta') \rangle. \qquad (8)$$

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

$$\|\hat{O} - TQ\|_\sigma^2 \le (1 + \epsilon)^2 \cdot \omega(\mathcal{F}) + C \cdot V_{\max}^2/(n \cdot \epsilon) + C' \cdot V_{\max} \cdot \delta + 2\beta \cdot G^2, \qquad (9)$$

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

$$\omega(\mathcal{F}) = \sup_{g \in \mathcal{F}} \inf_{f \in \mathcal{F}} \|f - Tg\|_\sigma. \qquad (10)$$

This result suggests the convergence rate of the previous result without PEER is maintained while PEER only adds a constant term $2\beta \cdot G^2$.

3.3. A toy example

We provide a toy example of a grid world (fig. 3a) to intuitively illustrate the functionality of PEER. In the grid world, the red state is the only state with a reward. We are interested in the Q values of the two gray states S_1 and S_2. Ideally, the policy is supposed to differentiate between those two adjacent states and accurately determine that S_1 has a higher value. We show the similarity between the representation of the Q-network and its target and the difference between the maximum Q value of two adjacent states in fig. 3. As shown in the fig. 3b, PEER is able to better distinguish the representation of the Q-network and its target compared to DQN. Thus, PEER is better equipped to differentiate between the maximum Q values of the two nearby states S_1 and S_2 (fig. 3c). Consequently, PEER yields a better policy (fig. 3d).

4. Experiments

We perform comprehensive experiments to thoroughly assess PEER. Specifically, we evaluate (i) **performance** by measuring average returns on multiple suites when combined with different backbone algorithms; (ii) **sample efficiency**
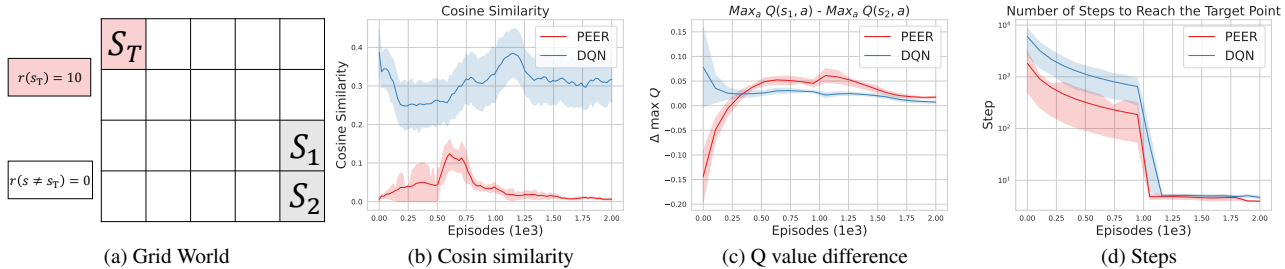
(a) Grid World   (b) Cosin similarity   (c) Q value difference   (d) Steps

Figure 3. Experiments on the grid world. (a) Grid world. We are interested in the values of S_1 and S_2. The policies are supposed to be able to differentiate the values of states S_1 and S_2 and tell that S_1 has the higher value. (b) The cosine similarity between the representation of the Q-network and its target. PEER (combined with DQN) effectively alleviates the similarity. (c) The difference between the max Q values of S_1 and S_2. Compared with DQN, PEER is able to better distinguish the Q values of two adjacent states by differentiating the representation of the Q-network and its target. (d) The number of steps to reach the S_T. PEER is better than DQN. The results are reported over five seeds and the shaded area represents a half standard deviation.

by comparing it with other algorithms at fixed timesteps; (iii) **compatibility**: whether PEER can be combined with other DRL algorithms such as off-policy methods TD3, contrastive unsupervised representation learning (CURL), and DrQ through extensive experiments; and (iv) the distinguishable representation property of PEER. To achieve this, we couple PEER with three representative DRL algorithms TD3 [28], CURL [6], and DrQ [9], and perform extensive experiments on four suites, namely, PyBullet, MuJoCo, DMControl, and Atari. PEER's simplicity and ease of implementation are noteworthy, requiring only one line of code. We deliberately avoid using any engineering tricks that could potentially enhance its performance. This decision is made to ensure that the reproducibility crisis, which has been a growing concern in the field of DRL [37, 38], is not further exacerbated. We also maintain a fixed newly introduced hyper-parameter β across all the experiments to achieve fair comparisons. Additional experimental details can be found in the Appendix.

4.1. Experimental settings

Hyper-parameters. We introduce only one additional hyper-parameter β to control the magnitude of regularization effectiveness of PEER. We report all results using a fixed value of $\beta = 5e - 4$. It is important to note that PEER may benefit from selecting a β value that is better suited to a specific environment.

Random seeds. To ensure the reproducibility of our experiments, we evaluate each tested algorithm using ten fixed random seeds unless otherwise specified. Moreover, we maintain fixed seeds across all experiments, including those used in PyTorch, Numpy, Gym, and CUDA packages.

Environments. To evaluate PEER, we use both state-based (state represented as vectors containing sensor information such as velocity, position, friction, etc) PyBullet and MuJoCo, and pixel-based (state represented as images) DM-

Control and Atari suites. The action space of PyBullet, MuJoCo, and DMControl is continuous, while that of Atari is discrete. Through the four experimental suites, we can check the performance and sample efficiency of PEER. We use the Gym [39] library for the interactive protocol. For PyBullet and MuJoCo suites, we run each tested algorithm for 1 million timesteps and evaluate the average return of the algorithm every 5k timesteps over ten episodes. For DMControl and Atari suites, following the commonly used experimental setting [6, 9], we measure the performance and sample-efficiency of tested algorithms at 100k and 500k environment timesteps, resulting in DMControl100k, DMControl500k, and Atari100k settings.

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

Table 1. The average return of the last ten evaluations over ten random seeds. PEER (coupled with TD3) outperforms all the compared algorithms, which shows that the PEER loss works in state-based environments. The best score is marked with colorbox. ± corresponds to a standard deviation over trials.

Baselines. We first evaluate PEER on the state-based PyBullet suite, using TD3, SAC [4], TRPO [40], PPO[41] as our baselines for their superior performance. And we couple PEER with TD3 on PyBullet and MuJoCo experiments. PEER works as preventing the similarity between the representation of the Q-network and its target. Another relevant baseline is MEPG [42], which enforces a dropout operator on both the Q-network and its target. Dropout operator [43, 44] is generally believed to prevent feature co-adaptation, which

500K Step Scores	State SAC	PlaNet	Dreamer	SAC+AE	DrQ	DrQ-v2	CURL	PEER
Finger, Spin	923 ± 21	561 ± 284	796 ± 183	884 ± 128	938 ± 103	789 ± 124	926 ± 45	864 ± 160
Cartpole, Swingup	848 ± 15	475 ± 71	762 ± 27	735 ± 63	868 ± 10	845 ± 18	841 ± 45	866 ± 17
Reacher, Easy	923 ± 24	210 ± 390	793 ± 164	627 ± 58	942 ± 71	748 ± 229	929 ± 44	980 ± 3
Cheetah, run	795 ± 30	305 ± 131	570 ± 253	550 ± 34	660 ± 96	607 ± 32	518 ± 28	732 ± 41
Walker, Walk	948 ± 54	351 ± 58	897 ± 49	847 ± 48	921 ± 45	696 ± 370	902 ± 43	946 ± 17
Ball_in_cup, Catch	974 ± 33	460 ± 380	879 ± 87	794 ± 58	963 ± 9	844 ± 174	959 ± 27	973 ± 5
100K Step Scores								
Finger, Spin	811 ± 46	136 ± 216	341 ± 70	740 ± 64	901 ± 104	325 ± 292	767 ± 56	820 ± 166
Cartpole, Swingup	835 ± 22	297 ± 39	326 ± 27	311 ± 11	759 ± 92	677 ± 214	582 ± 146	863 ± 17
Reacher, Easy	746 ± 25	20 ± 50	314 ± 155	274 ± 14	601 ± 213	256 ± 145	538 ± 233	961 ± 28
Cheetah, run	616 ± 18	138 ± 88	235 ± 137	267 ± 24	344 ± 67	273 ± 130	299 ± 48	499 ± 74
Walker, Walk	891 ± 82	224 ± 48	277 ± 12	394 ± 22	612 ± 164	171 ± 160	403 ± 24	714 ± 148
Ball_in_cup, Catch	746 ± 91	0 ± 0	246 ± 174	391 ± 82	913 ± 53	359 ± 228	769 ± 43	968 ± 7

Table 2. Scores achieved by PEER (coupled with CURL) on DMControl continuous control suite. PEER achieves superior performance on the majority (**9** out of **12**) tasks. And PEER also outperforms its backbone algorithm CURL on **11** out of **12** tasks by a large margin. The best score is marked with colorbox. ± corresponds to a standard deviation over trials.

is close to what PEER achieves. Therefore, we use MEPG combined with TD3 as a baseline, denoted as METD3. We employ the authors' TD3 implementation, along with the public implementation [45] for SAC, and the Baselines code-base [46] for TRPO and PPO. We opt to adhere to the authors' recommended default hyper-parameters for all considered algorithms.

Then we evaluate PEER on the pixel-based DMControl and Atari suites, combining it with CURL and DrQ algorithms. For DMControl, we select (i) PlaNet [5] and (ii) Dreamer [7], which learn a world model in latent space and execute planning; (iii) SAC+AE [8] uses VAE [47] and a regularized encoder; (iv) CURL using contrastive unsupervised learning to extract high-level features from raw pixels; (v) DrQ [9] and (vi) DrQ-v2 [18], which adopt data augmentation technique; and (vii) state-based SAC. For Atari suite, we select CURL, OTRainbow [48], Efficient Rainbow [49], Efficient DQN [9], and DrQ as baselines.

4.2. Results

PyBullet. We first evaluate the PEER loss in the state-based suite PyBullet. We show the final ten evaluation average return in table 1. The results show (i) PEER coupled with TD3 outperforms TD3 on all environments. Furthermore, (ii) PEER also surpasses all other tested algorithms. The superior performance of PEER shows that the PEER loss can improve the empirical performance of the off-policy model-free algorithm that does not adopt any representation learning techniques.

DMControl. We then perform experiments on the DM-Control suite. Specifically, we couple PEER with the CURL
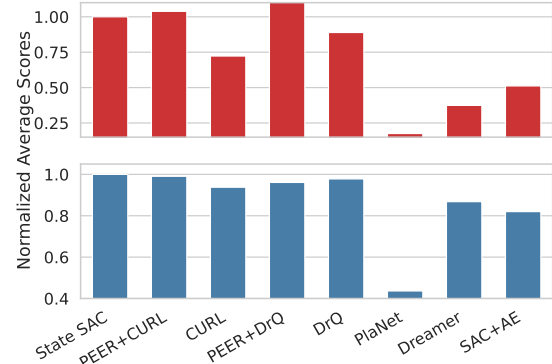

Figure 4. The normalized average scores on the DMControl suite. We normalize the average score of the tested algorithm by the average scores of State SAC. On the DMC100k benchmark, PEER (coupled with CURL and DrQ) outperforms all the compared algorithms including State SAC.

and DrQ algorithms, and run them in DMControl500k and DMControl100k settings. The results are shown in table 2 and fig. 4. The key findings are as follows: (i) On the DMControll500k setting, PEER coupled with CURL outperforms its backbone by a large margin on **5** out of **6** tasks, which shows the proposed PEER does improve the performance of its backbone algorithm. And the performance improvement on DMC500k shows that the PEER is beneficial for contrastive unsupervised learning (CURL). (ii) On the DMControl100k setting, The PEER (coupled with CURL) outperforms its backbone CURL on **6** out of **6** tasks. Besides, results in fig. 4 demonstrates that PEER (coupled with DrQ) improves the

Game	Human	Random	OTRainbow	Eff. Rainbow	Eff. DQN	DrQ	CURL	PEER$_{+CURL}$	PEER$_{+DrQ}$
Alien	7127.7	227.8	824.7	739.9	558.1	771.2	558.2	1218.9	712.7
Amidar	1719.5	5.8	82.8	188.6	63.7	102.8	142.1	185.2	163.1
Assault	742	222.4	351.9	431.2	589.5	452.4	600.6	631.2	721
Asterix	8503.3	210	628.5	470.8	341.9	603.5	734.5	834.5	918.2
BankHeist	753.1	14.2	182.1	51	74	168.9	131.6	78.6	12.7
BattleZone	37187.5	2360	4060.6	10124.6	4760.8	12954	14870	15727.3	5000
Boxing	12.1	0.1	2.5	0.2	-1.8	6	1.2	3.7	14.5
Breakout	30.5	1.7	9.8	1.9	7.3	16.1	4.9	3.9	8.5
ChopperCommand	7387.8	811	1033.3	861.8	624..4	780.3	1058.5	1451.8	1233.6
CrazyClimber	35829.4	10780.5	21327.8	16185.3	5430.6	20516.5	12146.5	18922.7	18154.5
DemonAttack	1971	152.1	711.8	508	403.5	1113.4	817.6	742.9	1236.7
Freeway	29.6	0	25	27.9	3.7	9.8	26.7	30.4	21.2
Frostbite	4334.7	65.2	231.6	866.8	202.9	331.1	1181.3	2151	537.4
Gopher	2412.5	257.6	778	349.5	320.8	636.3	669.3	583.6	681.8
Hero	30826.4	1027	6458.8	6857	2200.1	3736.3	6279.3	7499.9	3953.2
Jamesbond	302.8	29	112.3	301.6	133.2	236	471	414.1	213.6
Kangaroo	3035	52	605.4	779.3	448.6	940.6	872.5	1148.2	663.6
Krull	2665.5	1598	3277.9	2851.5	2999	4018.1	4229.6	4116.1	5444.7
KungFuMaster	22736.3	258.5	5722.2	14346.1	2020.9	9111	14307.8	15439.1	4090.9
MsPacman	6951.6	307.3	941.9	1204.1	872	960.5	1465.5	1768.4	1027.3
Pong	14.6	-20.7	1.3	-19.3	-19.4	-8.5	-16.5	-9.5	-18.2
PrivateEye	69571.3	24.9	100	97.8	351.3	-13.6	218.4	3207.7	8.2
Qbert	13455	163.9	509.3	1152.9	627.5	854.4	1042.4	2197.7	913.6
RoadRunner	7845	11.5	2696.7	9600	1491.9	8895.1	5661	10697.3	6900
Seaquest	42054.7	68.4	286.9	354.1	240.1	301.2	384.5	538.5	409.6
UpNDown	11693.2	533.4	2847.6	2877.4	2901.7	3180.8	2955.2	6813.5	7680.9

Table 3. Scores achieved by PEER (coupled with CURL and DrQ) and baselines on Atari. PEER achieves state-of-the-art performance on **19** out of **26** games. PEER implemented on top of CURL/DrQ improves over CURL/DrQ on **19/16** out of **26** games. Algorithms combined with PEER are reported across 10 random seeds. We also see that PEER achieves superhuman performance on Boxing, Freeway, JamesBond, Krull, and RoadRunner games. The best score is marked with colorbox.

sample efficiency of DrQ by a large margin. Overall, the PEER achieves SOTA performance on **11** out of **12** tasks. (iii) PEER outperforms State SAC on **6** out of **12** tasks. In fig. 4, we computed the average score of the tested algorithm on all environments under DMControl100k and DMControl500k settings, normalized by the score of State SAC. The results in DMControl100k show that PEER (combined with CURL and DrQ) is more sample-efficient than State SAC and other algorithms. And in the DMControl500k suite, the sample efficiency of PEER matches that of State SAC. These results illustrate that PEER remarkably improves the empirical **performance** and **sample efficiency** of the backbone algorithms TD3, CURL, and DrQ.

Atari. We present Atari100k experiments in table 3, which show that (i) PEER achieves state-of-the-art performance on **19** out of **26** environments in given timesteps. And (ii) PEER implemented on top of CURL/DrQ improves over CURL on **19/16** out of **26** games. (iii) We also see that PEER achieves superhuman performance on Boxing,

Freeway, JamesBond, Krull, and RoadRunner games. The empirical results demonstrate that PEER dramatically improves the **sample efficiency** of backbone algorithms.

4.3. Compatibility

Given a fixed hyper-parameter $\beta = 5e - 4$, PEER, as a plug-in, outperforms its backbone CURL algorithm on DMControl (**11** out of **12**) and Atari (**19** out of **26**) tasks, which shows that PEER is able to be incorporated into contrastive learn-based and data-augmentation methods to improve performance and sample efficiency. Besides, PEER coupled with DrQ also outperforms DrQ on **16** out of **26** environments. For state-based continuous control tasks, PEER (coupled with TD3) surpasses all the tested algorithms. These facts show that the compatibility of the PEER loss is remarkable and the PEER can be extended to incorporate more DRL algorithms. Theoretically, PEER tackles representation learning from the perspective of the representation of the Bellman equation, which is contrasting with other representa-

tion learning methods in DRL. Empirically, the performance of PEER loss coupled with representation learning DRL methods is better than that of backbone methods, which means that the PEER loss is orthogonal to other representation learning methods in DRL. Thus the compatibility of the PEER loss is auspicious.

4.4. PEER preserves distinguishable representation property

To validate whether the PEER regularizer preserves the distinguishable representation property or not, we measure the distinguishable representation discrepancy of the action value network and its target in PEER following section 3.1. We show the experimental results in fig. 1. The results show (i) TD3 and PEER (based on TD3) agents do enjoy the distinguishable representation property but that of PEER is more evident, which reveals the performance gain of PEER in this setting comes from better distinguishable representation property. (ii) The CURL agent does not maintain the distinguishable representation property on the tested DMControl suite, which negatively affects the model performance. And (iii) PEER also enjoys the distinguishable representation property on the pixel-based environment DMControl. Thus the performance of PEER is naturally improved due to the property being desirable. (iv) Combined with the performance improvement shown in performance experiments (section 4.2), PEER does improve the sample efficiency and performance by preserving the distinguishable representation property of the Q-network and its target.

5. Related Work

Representation learning [16, 50, 51, 52, 53, 54] aims at learning good or rich features of data. Such learned representation may be helpful for downstream tasks. The fields of natural language processing and computer vision have benefited from such techniques. It [6, 9, 16, 18, 19, 20] is widely acknowledged that good representations are conducive to improving the performance of DRL [4, 28, 35, 41, 42, 55, 56, 57]. Recent works [6, 9, 14, 15, 16, 17, 18] used self-supervised, unsupervised, contrastive representation learning, and data-augmentation approaches to improve the performance of DRL methods. There emerged several works studying representation learning from a geometric view [21, 22, 23]. Lyle et al. [24] explicitly considers representation capacity, which imposes regularization on the neural networks. The closest work to PEER is DR3 [58], which proposed a regularizer to tackle the implicit regularization effect of stochastic gradient descent in an offline RL setting. Despite the coincidental synergistic use of dot product form in PEER and DR3, we prove its necessity with different derivations and motivations. Our upper bound is rigorously derived for the representation of critic and its *target network*. But DR3 does not specify which network they

select. DR3 only works in the offline setting, while PEER is, in theory, applicable to both online and offline settings.

Our work differentiates from previous works from the following three perspectives. First, PEER tackles representation learning by preserving the distinguishable representation property instead of learning good representations with the help of auxiliary tasks. The convergence rate guarantee of PEER can be proved. Second, the experiments show that PEER is orthogonal to existing representation learning methods in DRL. Third, PEER is also suitable for environments based on both states and pixels while other representation learning methods in DRL are almost only performed on pixel-based suites.

6. Conclusion

In this work, we initiated an investigation of the representation property of the Q-network and its target and found that they ought to satisfy an inherently favorable distinguishable representation property. Then illustrative experimental results demonstrate that deep RL agents may not be able to preserve this desirable property during training. As a solution to maintain the distinguishable representation property of deep RL agents, we propose a straightforward yet effective regularizer, PEER, and provide a convergence rate guarantee. Our extensive empirical results indicate that, with a fixed hyper-parameter, PEER exhibits superior improvements across all suites. Furthermore, the observed performance improvements are attributable to the preservation of distinguishable representational properties. Previous work on value function representations often makes the representations of adjacent moments similar, which is thought to maintain some smoothing. However, our work demonstrates that there is an upper bound on this similarity and that minimizing it can preserve a beneficial property, leading to sampling efficiency and performance improvements. In some cases, the performance of PEER is commensurate with that of State SAC. However, we leave the task of further analyzing the reasons for this parity for future research. To the best of our knowledge, PEER is the first work to study the inherent representation property of Q-network and its target. We believe that our work sheds light on the nature of the inherent representational properties arising from combining the parameterization tool neural networks and RL.

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. 1

[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. 1, 2, 4

[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. 1

[4] 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. 1, 4, 5, 8

[5] 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. URL http://proceedings.mlr.press/v97/hafner19a.html. 1, 6

[6] 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. 1, 2, 3, 5, 8, 18

[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. URL https://openreview.net/forum?id=S1lOTC4tDS. 1, 6

[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. URL https://ojs.aaai.org/index.php/AAAI/article/view/17276. 1, 6

[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. URL https://openreview.net/forum?id=GY6-6sTvGaf. 1, 2, 5, 6, 8, 18

[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. URL https://proceedings.neurips.cc/paper/2021/hash/7f489f642a0ddb10272b5c31057f0663-Abstract.html. 1

[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. URL https://proceedings.neurips.cc/paper/2021/hash/099fe6b0b444c23836c4a5d07346082b-Abstract.html. 1

[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. 1

[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. URL https://openreview.net/forum?id=SJ6yPD5xg. 1

[14] 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. 1, 8

[15] 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. 1, 8

[16] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 1, 8

[17] 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. 1, 2, 8

[18] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=GY6-6sTvGaf. 1, 6, 8

[19] Dibya Ghosh and Marc G Bellemare. Representations for stable off-policy reinforcement learning. In *International Conference on Machine Learning*, pages 3556–3565. PMLR, 2020. 1, 8

[20] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e615c82aba461681ade82da2da38004a-Abstract.html. 1, 8

[21] 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. 1, 8

[22] 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. 1, 8

[23] Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. The information geometry of unsupervised reinforcement learning. *arXiv preprint arXiv:2110.02719*, 2021. 1, 8

[24] Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *Deep RL Workshop NeurIPS 2021*, 2021. 1, 8

[25] 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. URL https://proceedings.neurips.cc/paper/2017/hash/393c55aea738548df743a186d15f3bef-Abstract.html. 1, 2

[26] 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. URL https://ojs.aaai.org/index.php/AAAI/article/view/16880. 1, 2

[27] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 1, 2

[28] 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. 2, 3, 5, 8, 18

[29] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021. 2, 24

[30] 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. ISSN 2665-9638. doi: https://doi.org/10.1016/j.simpa.2020.100022. URL https://www.sciencedirect.com/science/article/pii/S2665963820300099. 2, 24

[31] 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. doi: 10.1109/IROS.2012.6386109. URL https://doi.org/10.1109/IROS.2012.6386109. 2, 24

[32] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013. 2, 24

[33] 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. 2

[34] 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. URL https://proceedings.neurips.cc/paper/2019/hash/3cf2559725a9fdfa602ec8c887440f32-Abstract.html. 2

[35] 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. 4, 8

[36] Daniel Rudolf. An upper bound of the minimal dispersion via delta covers. In *Contemporary Computational Mathematics-A Celebration of the 80th Birthday of Ian Sloan*, pages 1099–1108. Springer, 2018. 4

[37] 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. 5

[38] 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. 5

[39] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. 5

[40] 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. 5

[41] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 5, 8

[42] 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. 5, 8, 18

[43] David Warde-Farley, Ian J. Goodfellow, Aaron Courville, and Yoshua Bengio. An empirical analysis of dropout in piecewise linear networks. 2013. 5

[44] 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. 5

[45] Denis Yarats and Ilya Kostrikov. Soft actor-critic (sac) implementation in pytorch. https://github.com/denisyarats/pytorch_sac, 2020. 6, 18

[46] 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. 6, 18

[47] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016. 6

[48] Kacper Piotr Kielak. Do recent advancements in model-based deep reinforcement learning really improve data efficiency? In *URL https://openreview. net/forum*, page 6, 2020. 6

[49] Hado van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in 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 14322–14333, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1b742ae215adf18b75449c6e272fd92d-Abstract.html. 6

[50] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423. 8

[51] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020. 8

[52] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 8

[53] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 2020. URL http://proceedings.mlr.press/v119/chen20j.html. 8

[54] Olivier J. Hénaff. Data-efficient image recognition with contrastive predictive coding. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 4182–4192. PMLR, 2020. URL http://proceedings.mlr.press/v119/henaff20a.html. 8

[55] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016. 8

[56] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017. 8

[57] 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. 8

[58] Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. Dr3: Value-based deep reinforcement learning requires explicit regularization. In *International Conference on Learning Representations*, 2021. 8

[59] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010. 15

[60] 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. URL http://jmlr.org/papers/v22/20-1364.html. 18

[61] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR (Poster)*, 2015. 19, 20

[62] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W Ross. Randomized ensembled double q-learning: Learning fast without a model. In *International Conference on Learning Representations*, 2020. 30

[63] Nicklas Hansen, Hao Su, and Xiaolong Wang. Temporal difference learning for model predictive control. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 8387–8406. PMLR, 2022. URL https://proceedings.mlr.press/v162/hansen22a.html. 30