Is Exploration or Optimization the Problem for Deep Reinforcement Learning?

Glen Berseth
Université de Montréal, Mila - Quebec AI Institute, and CIFAR
glen.berseth@mila.quebec

Abstract

In the era of deep reinforcement learning, making progress is more complex, as the collected experience must be compressed into a deep model for future exploitation and sampling. Many papers have shown that training a deep learning policy under the changing state and action distribution leads to sub-optimal performance even collapse. This naturally leads to the concern that even if the community creates improved exploration algorithms or reward objectives, will those improvements fall on the *deaf ears* of optimization difficulties. This work proposes a new *practical* sub-optimality estimator to determine optimization limitations of deep reinforcement learning algorithms. Through experiments across environments and RL algorithms, it is shown that the difference between the best data generated is 2-3$\times$ better than the policies' learned performance. This large difference indicates that deep RL methods only exploit half of the good experience they generate.

1 Introduction

What is preventing deep reinforcement learning from solving harder tasks? Many papers have shown that training a deep learning policy under the changing state distribution (non-IID) leads to sub-optimal performance [Nikishin et al., 2022, Lyle et al., 2023, Dohare et al., 2024]. However, at a macro scale, it is not completely clear what causes these issues. Do the network and regularization changes from recent work improve exploration or exploitation, and which of these two issues is the larger concern to be addressed to advance deep RL algorithms? For example, better exploration algorithms can be created, but will the higher value experience fall on the *deaf ears* of the deep network optimization difficulties?

How can we understand if the limited deepRL performance is due to a lack of good exploration or deep network optimization (exploitation)? Normally in RL, to understand if there is a limitation, an oracle is needed to understand *sub-optimality*, how far the algorithm is from being optimal. However, that analysis is with respect to the best policy and aliases both causes of the limitations of either exploration or optimization. Instead, consider the example where a person is learning how to build good houses. There are two issues that may prevent the person from *consistently* building a high quality house: (1) they can't *explore* well enough to discover a good design or (2) they can explore well enough to find good designs, but they can't properly *exploit* their experience to replicate those good experience. For deep RL algorithms, which of these two issues is more prevalent?

To understand if exploration or exploitation is the larger culprit, a method is needed to estimate the *practical sub-optimality* between these cases. This estimator should (1) measure the agent's ability to explore, (2) while also estimating the average performance for the learning policy π^{θ}. While estimating the average policy performance is common, estimating the exploration ability for a policy is not. Extending the house-building metaphor, the idea is to estimate how close the agent ever got to constructing a good home. Therefore, to realize this estimator, we propose computing the *exploration* value for a policy that is calculated over prior experience, called the *experience optimal policy*. Using this concept, a new version of sub-optimality can be developed that can compute the difference between the *experience optimal policy* and the learned policy, shown in Figure 1a. If there is a large difference between these two, then the performance is limited by exploitation and optimization (model); however, if the difference between the *experience optimal policy* and the *learned policy* is small, then performance is limited by exploration (data).

The described estimator is used to better understand the reason deepRL algorithms do not solve certain *difficult* tasks. It is found that the limitation of deepRL agents in making progress on difficult tasks is not exploration but often exploitation. Therefore, this paper argues that to advance deep reinforcement learning research, further work is needed on optimization for exploitation under non-iid data. The proposed metric can serve multiple additional purposes. (1) For any RL practitioner, this metric can be used to quickly identify if the limitation in performance is an exploitation or exploration problem so that they can focus their efforts. (2) For the research community, this metric can be used across environments and algorithms to understand the performance of deepRL algorithms better and shed light on the *exploration* vs *exploitation* trade-off on a macro sense, to determine if to increase RL progress the community should be working more on exploitation problems[1]. (3) Showing that including exploration bonuses or scaling network size with RL algorithms increases the practical sub-optimality, indicating that optimization becomes a larger issue in that setting. Section 5 provides evidence to showcase these uses of this new view on sub-optimality, and finds for many environments the difference between the *experience optimal policy* and the learned policy to be larger than the difference between the learned policy and the initial policy. These findings suggest a significant exploitation issue and a need for improved optimization methods in RL.

2 Related Work

Since the first successes of RL and function approximation [Tesauro et al., 1995, Mnih et al., 2015, OpenAI et al., 2019], many recent works have shown great progress on integrating the complexities of deep learning and reinforcement learning [Hansen et al., 2022, van Hasselt et al., 2018]. Many have studied that certain model classes and loss assumptions make it easier to train more performant deepRL policies [Schwarzer et al., 2023, Farebrother et al., 2024], deepRL is even used to fine-tune the largest networks to create strong LLMs [OpenAI, 2022]. While deepRL is now being used across a growing number of applications, the broad limitations of current algorithms become less clear.

Deep Reinforcement Learning Training The field of methods to explain and improve on the limitations of combining function approximation and reinforcement learning (deepRL) is expanding. Much of the early work consisted of improving value-based methods to overcome training and non-IID data issues in DQN [Mnih et al., 2015] and DDPG [Lillicrap et al., 2015] and stochasticity [Schulman et al., 2015, 2017]. Recent adaptations improve over the initial algorithms that struggle with overestimation [van Hasselt et al., 2016, Bellemare et al., 2017, Hessel et al., 2018] or improving critic estimation [Fujimoto et al., 2018, Haarnoja et al., 2018, Lan et al., 2020, Kuznetsov et al., 2020, Chen et al., 2021]. The challenges in the space of learning policy are based on an unstable mix of function approximation, bootstrapping, and off-policy learning, called the Deadly Triad in DRL [van Hasselt et al., 2018, Achiam et al., 2019]. Many works focus on parts of triad, including: stabilizing effect of target network [Zhang et al., 2021b, Chen et al., 2022, Piché et al., 2022], difficulty of experience replay [Schaul et al., 2016, Kumar et al., 2020, Ostrovski et al., 2021], over-generalization [Ghiassian et al., 2020, Pan et al., 2021, Yang et al., 2022], representations in DRL [Zhang et al., 2021a, Li et al., 2022, Tang et al., 2022], off-policy correction [Nachum et al., 2019, Zhang et al., 2020a, Lee et al., 2021], interference [Cobbe et al., 2021, Raileanu and Fergus, 2021, Bengio et al., 2020] and architecture choices [Ota et al., 2020].

DeepRL Exploration Methods On top of the above training stability improvements is the desire to improve exploration by providing the agent with better signal to encourage exploration beyond just the extrinsic reward. These intrinsic rewards often compute some measure of state visitation or mutual information using a separate online learnt model. Count-based methods (curiosity) are early examples that encourage agents to cover a larger state space [Bellemare et al., 2016, Ostrovski et al.,

[1]This is not a judgement on the exploration community, in fact is with them in mind this work started so that their amazing research gets the best analysis it can, and great exploration algorithms are not misunderstood due to exploitation problems.

2017, Tang et al., 2017, Burda et al., 2018a], but they do not scale well to large state spaces. Several works [Pathak et al., 2017, Badia et al., 2020, Zhang et al., 2020b, 2021c] have built on curiosity frameworks to improve training and learning. However, it is not known how well RL algorithms will be able to learn from the additional experience.

DeepRL Scaling Methods Given the significant gains of using large models on many supervised learning problems, the RL community has been studying how to achieve similar gains from scale, but deep RL performance often drops when larger networks are used [Schwarzer et al., 2023, Tang and Berseth, 2024]. Recent works focus on network structure changes to avoid divergence and collapse, using normalization layers [Nauman et al., 2024, Lyle et al., 2024], regularization [Nikishin et al., 2022, Schwarzer et al., 2023, Galashov et al., 2024] or optimization adjustments [Lyle et al., 2024]. The goal in these prior works is to understand and improve performance when larger networks are used, but these papers are often limited to recovering prior performance, not understanding where RL in general is missing potential.

3 Background

In this section, a very brief review of the fundamental background of the proposed method is provided. reinforcement learning (RL) is formulated within the framework of an Markov Decision Processes (MDP) where at every time step t, the world (including the agent) exists in a state $\mathbf{s}_t \in \mathcal{S}$, where the agent is able to perform actions $\mathbf{a}_t \in \mathcal{A}$. The action to take is determined according to a policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$ which results in a new state $\mathbf{s}_{t+1} \in \mathcal{S}$ and reward $r_t = R(\mathbf{s}_t, \mathbf{a}_t)$ according to the transition probability function $P(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$. The policy is optimized to maximize the future discounted reward $\mathbb{E}_{r_0,\ldots,r_T}\left[\sum_{t=0}^{T}\gamma^t r_t\right]$, where T is the max time horizon, and γ is the discount factor. The formulation above generalizes to continuous states and actions. There are multiple RL algorithms that can be used to optimize the above objective. This work uses two of the most popular algorithms DQN [Mnih et al., 2015] and PPO [Schulman et al., 2017] to frame the challenges with optimizing and exploration.

Policy Gradient Definitions To discuss the difference between policy performance and estimators, it is useful to define the state visitation distribution $d^{\pi}_{s_0}(s)$ for a policy:

$$d^{\pi}_{s_0}(s) := (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \Pr(s_t = s|s_0), \tag{1}$$

where $\Pr^{\pi}(s_t = s|s_0)$ is the probability of the policy π visiting the future state s_t when starting from s_0. The policy gradient can be written in the form

$$\nabla_\theta V^{\pi_\theta}(s_0) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi_\theta}_{s_0}}\mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right]. \tag{2}$$

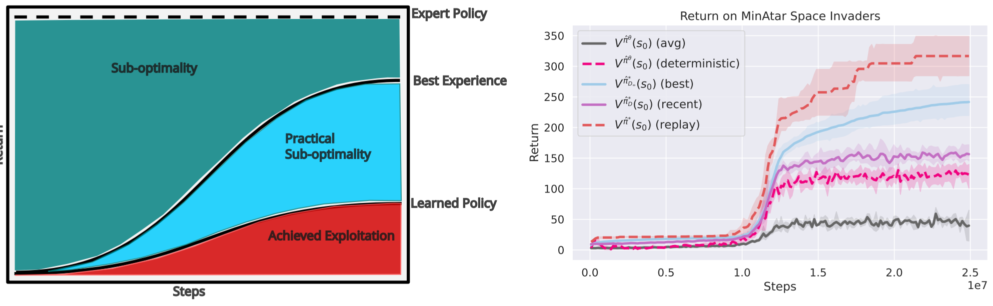
Then we can write out the **performance difference lemma** [Kakade and Langford, 2002] between two policies as

$$V^{\pi'}(s_0) - V^{\pi}(s_0) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi}_{s_0}}\mathbb{E}_{a\sim\pi(\cdot|s)}\left[A^{\pi'}(s,a)\right]. \tag{3}$$

Where $A^{\pi'}(s,a)$ is the advantage of policy π'.

4 Is Exploration or Exploitation the Issue for DeepRL?

Often, learning agents are concerned with the exploration vs exploitation trade-off. This trade-off is a helpful lens for discussing an agent's choices at a particular state $\mathbf{s}_t$, but this single state view focuses on *exploitation* as either: a type of greedy action selection, sampling from a learned policy, or utilizing a world model. However, in the age of deep learning and ever increasing model and data sizes, that lens misses the bigger picture of the definition *exploitation is making use of prior experience*, in that for each of these types of exploitation, there is a deep network θ optimization process over some experience $\mathcal{D}$ that is imperfect. However, it is not clear if the difference is from data distribution issues [Ostrovski et al., 2021] or optimization [Lyle et al., 2024]. To improve the understanding of the limitations of RL with function approximation (deepRL), we introduce estimators to quantify the difference between a policy's data-generating process (exploration/data) and its ability to learn from that data (exploitation/model).



(a) Example Exploitation sub-optimality Gap (b) MinAtar Space Invaders DQN

Figure 1: Left: Diagram of the practical sub-optimality = Best Experience - Learned Policy. On the right are results computing this exploitation gap as the difference between $V^{\hat{\pi}^*}(s_0)$ and $V^{\hat{\pi}^\theta}(s_0)$ in MinAtar SpaceInvaders.

In Figure 1 we show the conceptual version of studying this exploration vs exploitation problem, where the typical learning graph is now split into three sections: the performance of the average policy π^θ from *achieved exploitation* (red), which measures what that policy has learned, the potential performance, indicated by the optimal policy π^*, and a new estimator we call the *experience optimal* policy $\hat{\pi}^*$. The challenge is that π^θ can be arbitrarily bad compared to π^*, and normally it is not clear if the performance difference (Equation (3)) is because the agent is not exploring well ($V^{\hat{\pi}^*}(s_0)$ $<< V^{\pi^*}(s_0)$ and $V^{\hat{\pi}^\theta}(s_0) << V^{\pi^*}(s_0)$) or just not exploiting well ($V^{\hat{\pi}^\theta}(s_0) << V^{\pi^*}(s_0)$). This analysis can be particularly useful for evaluating exploration-focused algorithms. When evaluating the performance of a method, if only $V^{\hat{\pi}^\theta}(s_0)$ is considered, the analysis can miss the fact that the method is generating higher value experiences $V^{\hat{\pi}^*}(s_0)$, but the policy is not able to exploit them into θ properly. Therefore, to better understand reinforcement learning limitations, we introduce a new estimator for $\hat{\pi}^*$ to measure practical sub-optimality.

How to measure practical sub-optimality The optimal policy is defined as the policy that selects the best action at every state [Bellman, 1954]. Sub-optimality measures the difference between a policy's value $V^\pi(s)$ with respect to an optimal policy π^* with the value function $V^{\pi^*}(s)$. However, if the policy π can not explore optimally, using π^* is not very informative. Therefore, in addition to the theoretical optimal policy, we introduce the *experience optimal* policy $\hat{\pi}^*$ to represent the best policy the agent can achieve given the experience collected during training. If the environment is deterministic and the agent keeps a buffer of all prior experience D^∞, then,

$$\hat{\pi}^* = \underset{<a_0,\ldots,a_t>\in D^\infty}{\arg\max} \sum_{t=0}^{T} r(a_t, s_t) \tag{4}$$

This policy can also be understood as deterministically replaying the highest value sequence of actions $< a_0, \ldots, a_t >$ in the experience memory. This policy can be used to compute a new difference as the *exploitation sub-optimality* of the form $V^{\hat{\pi}^*}(s_0) - V^{\hat{\pi}^\theta}(s_0)$.

Most empirical works use $V^{\pi^\theta}(s_0)$ for comparing across algorithms to understand which algorithm performs the best on a set of tasks. While this information is helpful and enables the community to make steps forward in terms of performance, it does not provide information on why one algorithm is better than another. Consider the example where there are two algorithms A and B, algorithm A generates higher-value experience, but is not able to exploit them, and B does not generate higher-value experience, similar to its policy, but has been able to exploit that data well. Both A and B can have the same value $V^{\pi^A}(s_0) = V^{\pi^B}(s_0)$. Is A or B a better RL algorithm? In this work, we propose that B is the better algorithm, as it can properly exploit generated data. If the experience were equal, algorithm B would see the same experience as algorithm A, then B would result in better performance and have a smaller exploitation gap.

4.1 Softer Expert Estimators

While Equation (4) is a clear definition for computing an estimate of an optimal policy, it works best when the environment is completely deterministic. For example, the sequence of actions $a_0, \ldots, a_t$ can be replayed in the environment to reproduce $V^{\hat{\pi}^*}(s_0)$, however, this restrictive definition is less useful for non-deterministic environments where it is impossible for a stochastic policy/trajectory to outperform a deterministic one. Therefore, two additional methods are introduced to estimate the *potential* for the policy to learn from the data.

For the analysis, two versions of $V^{\hat{\pi}^*}(s_0)$ are introduced to approximate the performance on the *best experience*. For stochastic environments, the first version the best stocastic policy from the collected experience as top 5% of experience generated by the agent $V^{\hat{\pi}^*_{D_\infty}}(s_0)$, where D_∞ is all the experience collected by the agent. The second is the *recent* top 5% of data $V^{\hat{\pi}^*_{D}}(s_0)$ in the replay buffer D. To estimate the value function $V(s_0)$ from data, the sum of rewards the agent achieves in the environment, or the return, is used. The true value is computed using this function:

$$V^{\hat{\pi}^*}(s_0) = \frac{1}{k} \sum_{\tau \in D_{0:k}} \sum_{a_t, s_t \in \tau} R(a_t, s_t) \tag{5}$$

Where k is equal to $\frac{1}{20} \times |D|$ and D is sorted with the highest value trajectory starting at index 0.

The best *ever* and *recent* estimators both have their own reasoning. The best *ever* experience $V^{\hat{\pi}^*_{D_\infty}}(s_0)$ is a measure of how good the agent is at exploiting the best experience it ever generated. This notion is rather strong and difficult for any RL algorithm to match, as the agent may not currently have access to that experience for optimization, but it is a notion of lifetime achievement and represents a possible high-value policy and trajectory the agent could generate again. The *recent* best experience $V^{\hat{\pi}^*_{D}}(s_0)$ is a measure of the agent's ability to learn to match the best of the recent experience it has access to and can use for optimization. The *recent* notion can be more fair as it is possible for an agent to train on that experience to improve its performance actively, but as will be shown in Section 5.1, RL algorithms also struggle to match this performance.

4.2 For RL Algorithms

The above estimators can be used to understand the practical sub-optimality of an algorithm on an environment. That information is useful, but it does not speak about an algorithm holistically. For example, we may have the question, *how much does an algorithm suffer from exploitation limitations* and *which algorithms are the best at exploiting their generated data*? This information is paramount for the community to understand better where there is a larger benefit from time spent on research and development. To compute this information *across environments*, the estimator will need to be aggregated and normalized across environments.

To compute this aggregate estimator the upper bound from $V^{\hat{\pi}^*}(s_0)$ can be used in place of the less accurate and often overestimated the *optimalality gap* from *rliable* [Agarwal et al., 2021]. The gap computed using the proposed metric is also relative to the data the agent has generated, which can provide more rich signal than comparing the performance to some potential unattainable perfect agent. For example, when the optimal performance is not know a heuristic is often used to compute the return for the optimal policy by taking the max possible reward r_{max} and multiplying this by the inverse of the discount factor $V^{\pi^*}(s_0) \approx r_{\text{max}} * \frac{1}{1-\gamma}$. Instead, the proposed metric can be used to calculate the practical sub-optimality for an RL algorithm as:

$$\frac{1}{|\mathcal{T}|} \sum_{m \in \mathcal{T}} (V_m^{\hat{\pi}^*}(s_0) - V_m^{\pi^\theta}(s_0)) / (V_m^{\hat{\pi}^*}(s_0) - V_m^{\pi^0}(s_0)). \tag{6}$$

Where m is some task or environment. This metric is used in Section 5.4 to compare the aggregate weaknesses across RL algorithms.

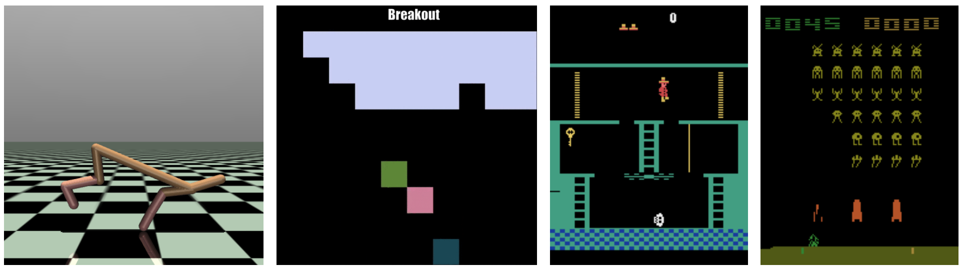
Implementation Details It is difficult to compute a general practical sub-optimality for any type of RL algorithm. On-policy algorithms do not keep around histories of recent data for evaluation, and off-policy algorithms don't track returns as they often use Q-functions for learning directly from rewards. To facilitate the tracking of these statistics, we introduce a wrapper that can be introduced into the RL algorithm code to track every reward, return, and end of episode. This wrapper is also used to compute the *best* $V^{\hat{\pi}^*_{D_\infty}}(s_0)$ and the *recent* $V^{\hat{\pi}^*_{D}}(s_0)$ version.

5 Experimental Results

In this section, the ability of practical sub-optimality for diagnosing learning issues is evaluated. This usefulness is determined in multiple ways: (1, Section 5.1) As a metric to determine the limitations of current RL algorithms on specific environments, (2) how recent methods for exploration or scaling increase or reduce the practical sub-optimality, and (3) The overall limitations of RL algorithms and if more exploration or exploitation is needed to improve performance over difficult/unsolved tasks.

Two popular RL algorithms are used for evaluation. First PPO [Schulman et al., 2017] is a common on-policy algorithm used for various problems, known for its ease of implementation and use. The other algorithm is DQN [Mnih et al., 2015], which is a popular RL algorithm for environments with discrete actions. These two algorithms cover the most common use cases for RL.

A selection of evaluation environments is included to cover a diverse range of the RL landscape. This diverse selection is important to understand better the practical sub-optimality there needs to be a difference between the generated data and the final policy's performance. Therefore, we focus on including experimental results on environments that are *difficult*. These difficult environments include using MinAtar [Young and Tian, 2019] and Atari [Bellemare et al., 2015, Aitchison et al., 2022] **SpaceInvaders**, **Asterix**, **LunarLander**, **Montezumas Revenge**, **Craftax**, and the **Atari-Five** [Aitchison et al., 2022]. We also include **Walker2d**, **HalfCheeta**, **Humanoid** as continuous action environments that are easier, and as will be shown, have little practical sub-optimality.



(a) HalfCheetah (b) MinAtar (c) MontezumaRevenge (d) Atari

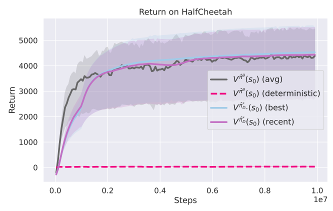
Figure 2: Environments used for evaluation include examples from Mujoco, MinAtar, and Atari.

To measure performance, we will look at the practical sub-optimality discussed in the previous section. In addition, the average return during learning is used to verify that the agents are learning, ensuring that the reason for the lack of practical sub-optimality is not due to the agent's inability to learn. All experiments are conducted over 4 random seeds.

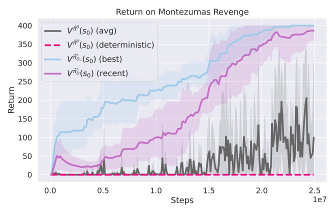
5.1 Per Task Sub-optimality

In this section, we can study which tasks express types of this practical sub-optimality, indicating a need for improvements in optimization over exploration. The first question (1) is whether tasks exhibit this type of gap, or if all tasks can be solved, or if policies can properly exploit the experience. In Figure 3a, we can see that for **HalfCheetah** there is little difference between $V^{\hat{\pi}^*_{D_\infty}}(s_0)$, $V^{\hat{\pi}^*_D}(s_0)$, and $V^{\hat{\pi}^\theta}(s_0)$, even though a high return is achieved; however, it is well known that **HalfCheetah** is no longer a difficult task for common RL algorithms. We can also see that the deterministic $\hat{\pi}^*$ poorly estimates the best performance in this non-deterministic environment and instead the softer versions work well. Examining tasks that are well-known to be difficult exploration problems reveals a different story. After training DQN on **Montezuma's Revenge** (Figure 3b), there is a surprisingly large gap where $V^{\hat{\pi}^\theta}(s_0)$ is noisy and near zero, yet the policy does generate many high-value trajectories, indicated by both a large difference between $V^{\hat{\pi}^*_{D_\infty}}(s_0)$ and $V^{\hat{\pi}^*_D}(s_0)$, but is not able to learn from these. These higher value trajectories are not rare. The $V^{\hat{\pi}^*_D}(s_0)$ line indicates that the policy, aside from a few spikes, is far from the best 5% of better experiences. We find similar results for many other environments and algorithms shown in Figure 3.

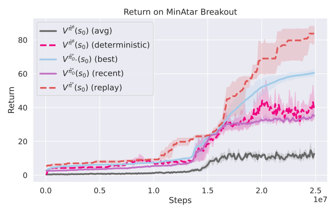
The practical sub-optimality may be an overestimate of true policy performance. To address this issue, we perform a pure analysis with a set of completely deterministic environments in Figure 1b and Figure 3c. Because these environments are *deterministic* settings, it is possible to compute a true



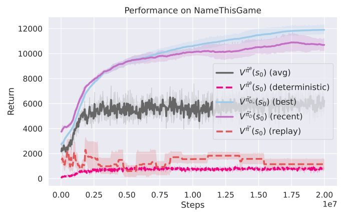
(a) PPO HalfCheetah



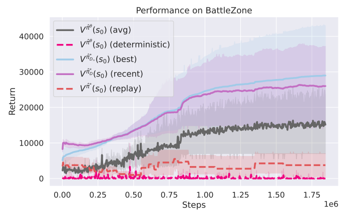
(b) PPO Montezums Revence



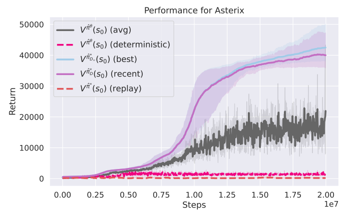
(c) MinAtar/Breakout DQN



(d) NameThisGame DQN



(e) BattleZone DQN



(f) Asterix DQN

Figure 3: Comparisons of different measures for global optimality and the learned policy π^θ. For environments with more complex exploration, such as Montezuma's Revenge, Breakout and SpaceInvaders, there is a large exploitation gap between $V^{\hat{\pi}^*_{D_\infty}}(s_0)$ and $V^{\hat{\pi}^\theta}(s_0)$.

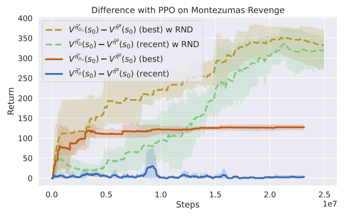
$V^{\hat{\pi}^*}(s_0)$ which is equal to the best single trajectory ever discovered. This best single trajectory is visualized as $V^{\hat{\pi}^*}(s_0)$, where the policy for $V^{\hat{\pi}^*}(s_0)$ is $a_0, \ldots, a_t$, which is replayed to visualize the score and indicate that to reach this performance, the policy θ needs to exploit this data well enough to reach that score. As can be seen, $V^{\hat{\pi}^*}(s_0) > V^{\hat{\pi}^*_{D_\infty}}(s_0) > V^{\hat{\pi}^\theta}(s_0)$, which indicates that $V^{\hat{\pi}^*_{D_\infty}}(s_0)$ may be slightly lower than the best performance, yet these trained policies struggle to produce the behaviour, indicating that often performance is limited by a lack of exploitation.

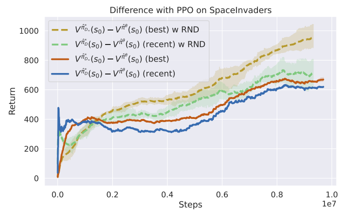
Last, to better understand these estimators for stochastic and deterministic settings, it is important to compare deterministic vs stochastic policy performance; in this case, the stochasticity added to the policy is causing a larger difference when the policy has learned a high-value behaviour. For PPO on continuous environments, this is equivalent to taking the mean of the policy, and for a discrete policy, the $\arg\max_a Q(s_t, a)$ is used. In Figure 3a the deterministic policy does poorly, this is likely because the agent quickly reaches states that are out of distribution, causing the agent to fail. Similar is true for **Montezuma's Revenge** with PPO. However, for MinAtar/Breakout and SpaceInvaders, the ϵ-greedy exploration of DQN knocks the policy off high-value paths, and the deterministic policy does well, even approaching $V^{\hat{\pi}^*_{D_\infty}}(s_0)$ for MinAtar/Breakout. We also see in Figure 1b and many other results that the difference does not decrease with additional training, indicating the gap is not the result of needing more experience or updates, but more significant changes to improve exploitation and optimization for deep learning.

5.2 Sub-optimality when adding exploration

This section asks the question *does adding exploration objectives increase the difference and therefore aggravate the optimization challenges*. This is analyzed by adding common exploration bonuses to the RL algorithms, RND [Burda et al., 2018b]. RND applies additional rewards to the extrinsic reward, encouraging the agent to explore a wider distribution of states, which should allow the agent to discover new, higher-reward states. These higher reward states should lead to larger returns, and if the algorithm is not effectively exploiting these rewards, a greater difference will result.

Figure 4 provides the results of the analysis of practical sub-optimality estimating Equation (3) compared with and without using RND. As we can see, the addition of RND improves the returns for DQN and PPO . However, the *difference* is also increased, indicating that as exploration is increased, so too are the issues of exploitation of experience in deep RL. This is an undesirable situation; as the agent improves its exploration, it actually learns less from the experience overall due to optimization issues.


(a) PPO MontezumasRevenge


(b) PPO SpaceInvaders

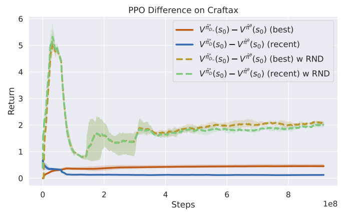
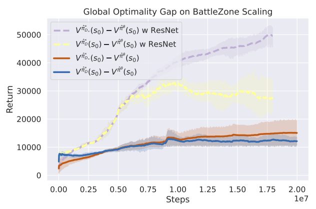
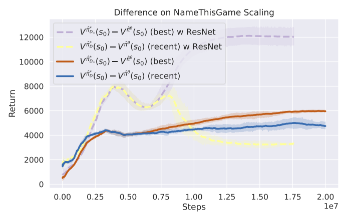

(c) PPO on Craftax

Figure 4: Comparisons of practical sub-optimality for best and recent performance compared to the average using Equation (3) with and without adding **RND**. These results show that with the addition of **RND**, the difference increases, indicating that adding exploration objectives is a double-edged sword, better exploration but more difficult exploitation.

5.3 Sub-optimality when Scaling Networks

Many recent reinforcement learning works are discovering improved algorithms' performance based on scaling networks [Schwarzer et al., 2023, Lyle et al., 2023, Obando-Ceron et al., 2024, Nauman et al., 2024, Tang and Berseth, 2024]. Are the challenges from scaling just optimization issues, or are these models also struggling to scale because the types of narrow distributions produced by larger models limit exploration? Two experiments were performed to investigate this question with networks of different sizes. First across Atari environments **BattleZone** and **NameThisGame** from the Atari-5 group [Aitchison et al., 2022] that is representative of the Full Atari Benchmark, and then across **HalfCheeta**. For the Atari environments, a comparison is made between training a policy that uses the normal C-51 network with a 3-layer CNN and using a ResNet. For the **HalfCheetah** environment, different numbers of layers are used between 4 and 256.


(a) DQN BattleZone


(b) DQN NameThisGame

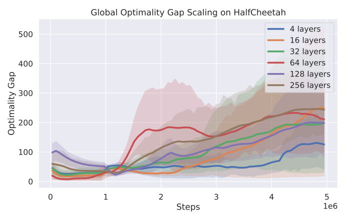

(c) PPO HalfCheetah

Figure 5: Comparisons of practical sub-optimality for models with different-sized networks. On the left and middle, it is shown that using a ResNet-18 instead of the common 3-layer CNN for BattleZone increased the difference. On the right, the difference for **HalfCheetah** increases as the number of layers is added, indicating increasing exploitation issues.

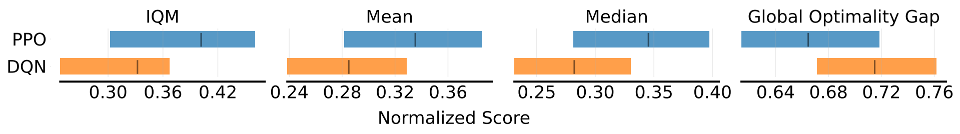
In Figure 5, the results of the described experiments are given. Interestingly, the results for the Atari environments show that the difference is much larger when the policy network is a ResNet-18 instead of a 3-layer CNN. This indicates two items: one, the policy is generating higher value trajectories, but it is not adequately learning from them, and two, the gap for $V^{\hat{\pi}^*_{D^\infty}}(s_0)$ and $V^{\hat{\pi}^*_D}(s_0)$ is very close, indicating that the policy is struggling to match these higher value experiences even when they are in the current replay buffer. With **HalfCheetah**, the issue of scale is studied by training a policy over networks of 6 different sizes. In Figure 5c, the $V^{\hat{\pi}^*_{D^\infty}}(s_0) - V^{\hat{\pi}^\theta}(s_0)$ is shown, and there is a trend that as the number of layers increases, the practical sub-optimality increases. This is interesting because in Figure 3a the performance with one layer is given and there is no gap. The introduction of additional layers quickly introduces exploitation issues, keeping the policy from learning the same performance in Figure 3a. This collective information suggests that scaling networks does not likely cause exploration issues, but rather reinforces the commonly understood cause of exploitation (optimization/model) issues with scale.

5.4 Algorithm Sub-optimality

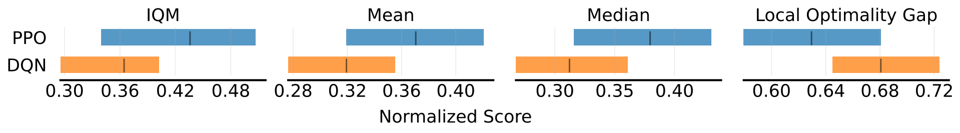
Is algorithm progress limited by weaknesses in exploration or exploitation? This question can be estimated by using the practical sub-optimality to compare aggregate analysis across tasks and

RL algorithms, as described in Section 4.2. Starting with aggregate analysis across the *AtariFive* environments, we can see in Figure 6a that DQN and PPO are only able to achieve a little over 30% of the performance of their best experience (lower is better). This high value indicates that both of these algorithms struggle to produce the best possible results they have experienced. In this case, $V^{\hat{\pi}^*_{D_\infty}}(s_0)$ (Figure 6a) is similar to $V^{\hat{\pi}^*_D}(s_0)$ (Figure 6b), indicating that the RL algorithms are experiencing high returns regularly, with a value of 0.68, they are not sufficiently capturing.

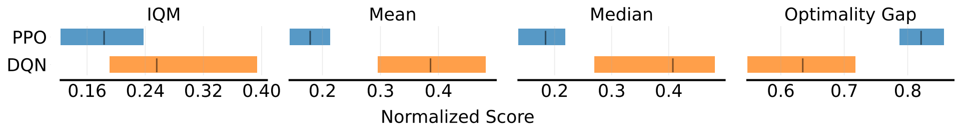
Interestingly and conversely, the *rliable* optimality gap indicates that DQN is better than PPO in Figure 6c, because DQN does achieve higher average policy performance, but the analysis from comparing to $V^{\hat{\pi}^*_D}(s_0)$, in Figure 6b shows us that even though DQN performs better than PPO, DQN is still generating a lot of high-value experience that it is not able to exploit. Conversely, because PPO is performing worse according to *rliable*, but has a better $V^{\hat{\pi}^*_D}(s_0) - V^{\hat{\pi}^\theta}(s_0)$, improved exploration would improve PPO more than it would DQN. Overall, these results suggest that both algorithms struggle to extract the most from their experience and that *rliable* is not telling the full story.



(a) $V^{\hat{\pi}^*_{D_\infty}}(s_0) - V^{\hat{\pi}^\theta}(s_0)$ across Atari-5 environments.



(b) $V^{\hat{\pi}^*_D}(s_0) - V^{\hat{\pi}^\theta}(s_0)$ across Atari-5 environments.



(c) Normal *rliable* evaluation across Atari-5 environment.

Figure 6: *rliable* plots for **PPO** and **DQN** over all environments in this paper. This measure gives an aggregate view for each algorithm as each sample is normalized using the practical sub-optimality from each run's generated data.

6 Discussion

This work has introduced a method to study the limitations of deep RL algorithms in the space of optimization and exploitation challenges. An estimator is introduced to support this position. The estimator is used to show that common RL algorithms struggle to exploit their experience and that adding exploration bonuses and scaling networks exacerbates these issues. This estimator can be used to assist users in understanding if poor performance in an environment is the result of limited exploration (data problem) or more stable optimization to make progress (model problem). Because RL agents collect vastly different data during training, it can be difficult to compare performance across algorithms. This estimator adjusts the comparison to show how well the algorithm did compared to the distribution of collected data (experience). Because the estimator comparison over generated experience measures the sub-optimality relative to the agent's generated experience, it can be better suited to task-independent comparisons. In the future, this metric can be used to evaluate broadly across produced algorithms to assist researchers and practitioners in their analysis.

References

J. Achiam, E. Knight, and P. Abbeel. Towards characterizing divergence in deep q-learning. *arXiv preprint*, arXiv:1903.08894, 2019. 2

Rishabh Agarwal, Max Schwarzer, P S Castro, Aaron C Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Neural Inf Process Syst*, 34:29304–29320, August 2021. 5

Matthew Aitchison, Penny Sweetser, and Marcus Hautter. Atari-5: Distilling the arcade learning environment down to five games. *ICML*, abs/2210.02019:421–438, October 2022. 6, 8

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies, 2020. 3

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016. 2

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 4148–4152. AAAI Press, 2015. ISBN 978-1-57735-738-4. URL `http://dl.acm.org/citation.cfm?id=2832747.2832830`. 6

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *ICML*, 2017. 2

Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954. 4

Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning. In *ICML*, 2020. 2

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation, 2018a. 3

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *ICLR*, 2018b. 7

Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. Randomized ensembled double q-learning: Learning fast without a model. In *ICLR*, 2021. 2

Zaiwei Chen, John-Paul Clarke, and Siva Theja Maguluri. Target network and truncation overcome the deadly triad in q-learning. *arXiv preprint*, arXiv:2203.02628, 2022. 2

Karl Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *ICML*, 2021. 2

Shibhansh Dohare, J Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A Rupam Mahmood, and Richard S Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, August 2024. 1

Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, Aviral Kumar, and Rishabh Agarwal. Stop regressing: Training value functions via classification for scalable deep RL. *arXiv [cs.LG]*, March 2024. 2

S. Fujimoto, H. v. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *ICML*, 2018. 2

Alexandre Galashov, Michalis K Titsias, Andr'as Gyorgy, Clare Lyle, Razvan Pascanu, Y W Teh, and M Sahani. Non-stationary learning of neural networks with automatic soft parameter reset. *Neural Inf Process Syst*, abs/2411.04034:83197–83234, November 2024. 3

Sina Ghiassian, Banafsheh Rafiee, Yat Long Lo, and Adam White. Improving performance in reinforcement learning by breaking generalization in neural networks. In *AAMAS*, 2020. 2

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018. 2

Nicklas Hansen, Xiaolong Wang, and H Su. Temporal difference learning for model predictive control. *ICML*, 162:8387–8406, March 2022. 2

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018. 2

Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the nineteenth international conference on machine learning*, pages 267–274, 2002. 3

Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. In *NeurIPS*, 2020. 2

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry P. Vetrov. Controlling overestimation bias with truncated mixture of continuous distributional quantile critics. In *ICML*, 2020. 2

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. In *ICLR*, 2020. 2

Jongmin Lee, Wonseok Jeon, Byung-Jun Lee, Joelle Pineau, and Kee-Eung Kim. Optidice: Offline policy optimization via stationary distribution correction estimation. In *ICML*, 2021. 2

Boyan Li, Hongyao Tang, Yan Zheng, Jianye Hao, Pengyi Li, Zhen Wang, Zhaopeng Meng, and Li Wang. Hyar: Addressing discrete-continuous action reinforcement learning via hybrid action representation. In *ICLR*, 2022. 2

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2015. 2

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Ávila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *ICML*, 2023. 1, 8

Clare Lyle, Zeyu Zheng, Khimya Khetarpal, James Martens, H V Hasselt, Razvan Pascanu, and Will Dabney. Normalization and effective learning rates in reinforcement learning. *Neural Inf Process Syst*, abs/2407.01800:106440–106473, July 2024. 3

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. 2, 3, 6

Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience. *arXiv preprint*, arXiv:1912.02074, 2019. 2

Michal Nauman, Mateusz Ostaszewski, Krzysztof Jankowski, Piotr Miłoś, and Marek Cygan. Bigger, regularized, optimistic: scaling for compute and sample-efficient continuous control. *arXiv preprint arXiv:2405.16158*, 2024. 3, 8

E. Nikishin, M. Schwarzer, P. D'Oro, P. Bacon, and A. C. Courville. The primacy bias in deep reinforcement learning. In *ICML*, Proceedings of Machine Learning Research, 2022. 1, 3

Johan Obando-Ceron, Ghada Sokar, Timon Willi, Clare Lyle, Jesse Farebrother, Jakob N Foerster, G Dziugaite, D Precup, and Pablo Samuel Castro. Mixtures of experts unlock parameter scaling for deep RL. *ICML*, abs/2402.08609, February 2024. 8

OpenAI. Chatgpt: Optimizing language models for dialogue, 2022. URL `https://openai.com/blog/chatgpt/`. 2

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *arXiv [cs.LG]*, December 2019. 2

Georg Ostrovski, Marc G. Bellemare, Aaron van den Oord, and Remi Munos. Count-based exploration with neural density models, 2017. 2

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. In *NeurIPS*, 2021. 2, 3

Kei Ota, Tomoaki Oiki, Devesh K. Jha, Toshisada Mariyama, and Daniel Nikovski. Can increasing input dimensionality improve deep reinforcement learning? In *ICML*, 2020. 2

Yangchen Pan, Kirby Banman, and Martha White. Fuzzy tiling activations: A simple approach to learning sparse representations online. In *ICLR*, 2021. 2

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017. 3

Alexandre Piché, Valentin Thomas, Joseph Marino, Rafael Pardinas, Gian Maria Marconi, Christopher Pal, and Mohammad Emtiyaz Khan. Bridging the gap between target networks and functional regularization. *Transactions on Machine Learning Research*, 2022. 2

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *ICML*, 2021. 2

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *ICLR*, 2016. 2

J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, 2015. 2

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, arXiv:1707.06347, 2017. 2, 3, 6

Max Schwarzer, Johan Samir Obando-Ceron, Aaron C. Courville, Marc G. Bellemare, Rishabh Agarwal, and Pablo Samuel Castro. Bigger, better, faster: Human-level atari with human-level efficiency. In *ICML*, 2023. 2, 3, 8

H. Tang, Z. Meng, J. Hao, C. Chen, D. Graves, D. Li, C. Yu, H. Mao, W. Liu, Y. Yang, W. Tao, and L. Wang. What about inputting policy in value function: Policy representation and policy-extended value function approximator. In *AAAI*, 2022. 2

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration: A study of count-based exploration for deep reinforcement learning, 2017. 3

Hongyao Tang and Glen Berseth. Improving deep reinforcement learning by reducing the chain effect of value and policy churn. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL `https://openreview.net/forum?id=cQoAgPBARc`. 3, 8

Gerald Tesauro et al. Temporal difference learning and td-gammon. *Communications of the ACM*, 38 (3):58–68, 1995. 2

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016. 2

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint*, arXiv:1812.02648, 2018. 2

Ge Yang, Anurag Ajay, and Pulkit Agrawal. Overcoming the spectral bias of neural value approximation. In *ICLR*, 2022. 2

Kenny Young and Tian Tian. MinAtar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv [cs.LG]*, March 2019. 6

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *ICLR*, 2021a. 2

Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values. In *ICML*, 2020a. 2

Shangtong Zhang, Hengshuai Yao, and Shimon Whiteson. Breaking the deadly triad with a target network. In *ICML*, 2021b. 2

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. Bebold: Exploration beyond the boundary of explored regions. *arXiv preprint arXiv:2012.08621*, 2020b. 3

Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. Noveld: A simple yet effective exploration criterion. *Advances in Neural Information Processing Systems*, 34, 2021c. 3